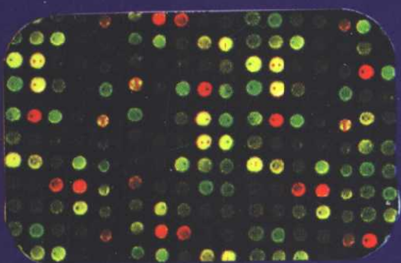
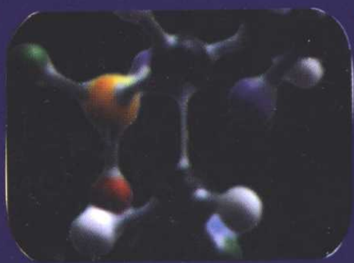
Current Topics in Computational Molecular Biology

计算分子生物学前沿课题

Edited by Tao Jiang Ying Xu Michael Q. Zhang


清华大学出版社
http://www.tup.tsinghua.edu.cn

The MIT Press
http://mitpress.mit.edu

VISIT…

LANZAROTE Caliente.com

Current Topics in Computational Molecular Biology

计算分子生物学前沿课题

Edited by

Tao Jiang
Ying Xu
Michael Q. Zhang

江苏工业学院图书馆
藏书章

Tsinghua University Press   The MIT Press

内容简介

本书概观综述了计算生物学的传统课题，如蛋白质结构模拟计算和基因序列比较，以及目前的发展热点，如基因表达数据分析和比较基因学，并阐述了算法、统计，数据库和一些以人工智能为基础的技术在解决生物问题上的应用。本书还包括了对计算生物学基础知识介绍，具体实用统计模型和分子生物学的计算方法实例。本书每章节围绕一具体议题阐述，自成一体，以便阅读。

© 2002 Tsinghua University Press and Massachusetts Institute of Technology

All rights reserved. No part of this book may be reproduced in any form by any electronic or mechanical means (including photocopying, recording, or information storage and retrieval) without permission in writing from the publisher.

Published in association with Tsinghua University Press, Beijing, China, as part of TUP's Frontiers of Science and Technology for the 21st Century Series.

This book was set in Times New Roman on 3B2 by Asco Typesetters, Hong Kong and was printed and bound in the United States of America.

版权所有，翻印必究。

（以下内容为在中国印制部分）

书　名：计算分子生物学前沿课题

作　者：Tao Jiang　Ying Xu　Michael Q. Zhang

出版者：清华大学出版社（北京清华大学学研大厦，邮编 100084）

http://www.tup.tsinghua.edu.cn

印刷者：清华大学印刷厂

发行者：新华书店总店北京发行所

开　本：787×960　1/16　印张：35

版　次：2002 年 4 月第 1 版　2003 年 4 月第 2 次印刷

书　号：ISBN 7-302-05378-2

ISBN 0-262-10092-4

印　数：801～1300（中国印制部分）

定　价：70.00 元

21世纪科技前沿丛书
Frontiers of Science and Technology for the 21st Century

主　编	刘国奎	美国阿冈国家实验室
Editor in Chief	G. K. Liu	Argonne National Laboratory, USA
编　委	范波涛	法国巴黎第七大学
Board of Editors	B. T. Fan	University of Paris-Ⅶ, France
	时东陆	美国辛辛那提大学
	D. L. Shi	University of Cincinnati, USA
	王中林	美国乔治亚理工学院
	Z. L. Wang	Georgia Institute of Technologies, USA
	韦大同	美国朗讯科技公司
	D. T. Wei	Lucent Technologies, USA
	徐　鹰	美国橡树岭国家实验室
	Y. Xu	Oak Ridge National Laboratory, USA
	章效锋	美国劳伦斯伯克莱国家实验室
	X. F. Zhang	Lawrence Berkeley National Laboratory, USA
	张　泽	中国科学院北京电子显微镜实验室，物理研究所
	Z. Zhang	Beijing Laboratory of Electron Microscopy, Institute of Physics Chinese Academy of Sciences, China

《21 世纪科技前沿》

丛书序言

由清华大学出版社出版的这套丛书是基础科学和应用科学领域内的专门著作。除了可作为研究生教材外,也可作为科研和工程技术人员的参考书。在丛书的题材选择中,着重考虑目前比较活跃而且具有发展前景的新兴学科。因此,这套丛书大都涉及交叉和新兴学科的内容。编写的方式大多由主编策划并组织本学科有影响的专家共同执笔完成,从而使每一本书的系统性和各章节内容的连贯性得到了充分的兼顾。丛书涵盖学科的最新学术进展,兼顾到基本理论和新技术、新方法的介绍,并引入必要的导论和充分的参考文献以适应具有不同学术背景的读者。编撰一套容纳多学科的科技丛书是一项浩繁的工作,我们希望通过主编和作者的集体努力和精诚协作,使整套丛书的学术水准能够保持在较高的水平上。

编辑《21 世纪科技前沿》丛书是由"旅美中国科学家工程师协会"发起的一项国际科技界的合作。传递信息,加强交流,促进新世纪的科技繁荣是编著者们参与此项工作的共同信念。此外,这套丛书还具有特别的纪念意义。20 年前,历史的进程使成千上万的中国学生、学者有机会走出国门,到世界各地学习和从事科学研究。今天,活跃在世界科技前沿领域的中华学子们没有忘记振兴祖国科技教育事业的责任和推动国际学术交流与合作的义务。正是基于这一共同的心愿,大家积极参与这套系列丛书的撰写、组稿和编辑工作。为此,我们愿以这套丛书来纪念中国改革开放 20 周年。

编委会
1999. 6

《Frontiers of Science and Technology for the 21st Century》
FOREWORD

Over the next several years, Tsinghua University Press will publish a series of books addressing progress in basic sciences and innovations in technology. We have made no attempt to pursue a comprehensive coverage of all disciplines of science and technology. Rather, topics for this series were selected with an emphasis on the currently active forefront of science and technology that will be contemporary in the next century. Most books in this series will deal with subjects of cross disciplines and newly emerging fields. Each book will be completed by individual authors or in a collaborative effort managed by an editor(s), and will be self-consistent, with contents systematically focused on review of the most recent advances and description of current progresses in the field. Sufficient introduction and references will be provided for readers with varying backgrounds. We have realize clearly the challenge of encompassing the diverse subjects of science and technology in one series. However, we hope that, through intensive collaboration between the authors and editors, high standards in editorial quality and scientific merit will be maintained for the entire series.

The international collaboration on this series has been coordinated by the Association of Chinese Scientists and Engineers-USA(ACSE). In the science community, authors voluntarily publish their results and discoveries in the full

conviction that science should serve human society. The editors and authors of this series share this academic tradition, and many of them are fulfilling a spiritual commitment as well. For our editors and authors who were graduated from universities in China and further educated abroad in science and engineering, this is an opportunity to dedicate their work to the international education community and to commemorate the historical open-door movement that began in China two decades ago. When the human society enters the information age, there is no geographic boundary for science. The Editorial committee hopes that this series will promote further international collaboration in scientific research and education at the dawn of the new century.

The Editorial Committee
1999. 6

Current Topics in Computational Molecular Biology

Science is advanced by new observations and technologies. The Human Genome Project has led to a massive outpouring of genomic data, which has in turn fueled the rapid developments of high-throughput biotechnologies. We are witnessing a revolution driven by the high-throughput biotechnologies and data, a revolution that is transforming the entire biomedical research field into a new systems level of genomics, transcriptomics, and proteomics, fundamentally changing how biological science and medical research are done. This revolution would not have been possible if there had not been a parallel emergence of the new field of computational molecular biology, or bioinformatics, as many people would call it. Computational molecular biology/bioinformatics is interdisciplinary by nature and calls upon expertise in many different disciplines—biology, mathematics, statistics, physics, chemistry, computer science, and engineering; and is ubiquitous at the heart of all large-scale and high-throughput biotechnologies. Though, like many emerging interdisciplinary fields, it has not yet found its own natural home department within traditional university settings, it has been identified as one of the top strategic growing areas throughout academic as well as industrial institutions because of its vital role in genomics and proteomics, and its profound impact on health and medicine.

At the eve of the completion of the human genome sequencing and annotation, we believe it would be very useful and timely to bring out this up-to-date survey of current topics in computational molecular biology. Because this is a rapidly developing field and covers a very wide range of topics, it is extremely difficult for any individual to write a comprehensive book. We are fortunate to be able to pull together a team of renowned experts who have been actively working at the forefront of each major area of the field. This book covers most of the important topics in computational molecular biology, ranging from traditional ones such as protein structure modeling and sequence alignment, to the recently emerged ones such as expression data analysis and comparative genomics. It also contains a general introduction to the field, as well as a chapter on general statistical modeling and computational techniques in molecular biology. Although there are already several books on computational molecular biology/bioinformatics, we believe that this book is unique as it covers a wide spectrum of topics (including a number of new ones not covered in existing books, such as gene expression analysis and pathway databases) and it combines algorithmic, statistical, database, and AI-based methods for biological problems.

Although we have tried to organize the chapters in a logical order, each chapter is a self-contained review of a specific subject. It typically starts with a brief overview of a particular subject, then describes in detail the computational techniques used and the computational results generated, and ends with open challenges. Hence the reader need not read the chapters sequentially. We have selected the topics carefully so that

the book would be useful to a broad readership, including students, nonprofessionals, and bioinformatic experts who want to brush up topics related to their own research areas.

The 19 chapters are grouped into four sections. The introductory section is a chapter by Temple Smith, who attempts to set bioinformatics into a useful historical context. For over half a century, mathematics and even computer-based analyses have played a fundamental role in bringing our biological understanding to its current level. To a very large extent, what is new is the type and sheer volume of new data. The birth of bioinformatics was a direct result of this new data explosion. As this interdisciplinary area matures, it is providing the data and computational support for functional genomics, which is defined as the research domain focused on linking the behavior of cells, organisms, and populations to the information encoded in the genomes.

The second of the four sections consists of six chapters on computational methods for comparative sequence and genome analyses.

Liu's chapter presents a systematic development of the basic Bayesian methods alongside contrasting classical statistics procedures, emphasizing the conceptual importance of statistical modeling and the coherent nature of the Bayesian methodology. The missing data formulation is singled out as a constructive framework to help one build comprehensive Bayesian models and design efficient computational strategies. Liu describes the powerful computational techniques needed in Bayesian analysis, including the expectation-maximization algorithm for finding the marginal mode, Markov chain Monte Carlo algorithms for simulating from complex posterior distributions, and dynamic programming-like recursive procedures for marginalizing out uninteresting parameters or missing data. Liu shows that the popular motif sampler used for finding gene regulatory binding motifs and for aligning subtle protein motifs can be derived easily from a Bayesian missing data formulation.

Huang's chapter focuses on methods for comparing two sequences and their applications in the analysis of DNA and protein sequences. He presents a global alignment algorithm for comparing two sequences that are entirely similar. He also describes a local alignment algorithm for comparing sequences that contain locally similar regions. The chapter gives efficient computational techniques for comparing two long sequences and comparing two sets of sequences, and it provides real applications to illustrate the usefulness of sequence alignment programs in the analysis of DNA and protein sequences.

The chapter by Jiang and Wang provides a survey on computational methods for multiple sequence alignment, which is a fundamental and challenging problem in computational molecular biology. Algorithms for multiple sequence alignment are routinely used to find conserved regions in biomolecular sequences, to construct

family and superfamily representations of sequences, and to reveal evolutionary histories of species (or genes). The authors discuss some of the most popular mathematical models for multiple sequence alignment and efficient approximation algorithms for computing optimal multiple alignment under these models. The main focus of the chapter is on recent advances in combinatorial (as opposed to stochastic) algorithms.

Kearney's chapter illustrates the basic concepts in phylogenetics, the design and development of computational tools for evolutionary analyses, using the quartet method as an example. Quartet methods have recently received much attention in the research community. This chapter begins by examining the mathematical, computational, and biological foundations of the quartet method. A survey of the major contributions to the method reveals an excess of diverse and interesting concepts indicative of a ripening research topic. These contributions are examined critically with strengths, weakness, and open problems.

Sankoff and El-Mabrouk's chapter describes the basic concepts of genome rearrangement and applications. Genome structure evolves through a number of non-local rearrangement processes that may involve an arbitrarily large proportion of a chromosome. The formal analysis of rearrangements differs greatly from DNA and protein comparison algorithms. In this chapter, the authors formalize the notion of a genome in terms of a set of chromosomes, each consisting of an ordered set of genes. The chapter surveys genomic distance problems, including the Hannenhalli-Pevzner theory for reversals and translocations, and covers the progress to date on phylogenetic extensions of rearrangement analysis. Recent work focuses on problems of gene and genome duplication and their implications for genomic distance and genome-based phylogeny.

The chapter by Li describes the author's work on compressing DNA sequences and applications. The chapter concentrates on two programs the author has developed: a lossless compression algorithm, GenCompress, which achieves the best compression ratios for benchmark sequences; and an entropy estimation program, GTAC, which achieves the lowest entropy estimation for benchmark DNA sequences. The author then discusses a new information-based distance measure between two sequences and shows how to use the compression programs as heuristics to realize such distance measures. Some experiments are described to demonstrate how such a theory can be used to compare genomes.

The third section covers computational methods for mining biological data and discovering patterns hidden in the data.

The chapter by Xu presents an overview of the major statistical techniques for quantitative trait analysis. Quantitative traits are defined as traits that have a con-

tinuous phenotypic distribution. Variances of these traits are often controlled by the segregation of multiple loci plus an environmental variance. Localization of these quantitative trait loci (QTL) on the chromosomes and estimation of their effects using molecular markers are called QTL linkage analysis or QTL mapping. Results of QTL mapping can help molecular biologists target particular chromosomal regions and eventually clone genes of functional importance.

The chapter by Solovyev describes statistically based methods for the recognition of eukaryotic genes. Computational gene identification is an issue of vital importance as a tool of identifying biologically relevant features (protein coding sequences), which often cannot be found by the traditional sequence database searching technique. Solovyev reviews the structure and significant characteristics of gene components, and discusses recent advances and open problems in gene-finding methodology and its application to sequence annotation of long genomic sequences.

Zhang's chapter gives an overview of computational methods currently used for identifying eukaryotic PolII promoter elements and the transcriptional start sites. Promoters are very important genetic elements. A PolII promoter generally resides in the upstream region of each gene; it controls and regulates the transcription of the downstream gene.

In their chapter, Shamir and Sharan describe some of the main algorithmic approaches to clustering gene expression data, and briefly discuss some of their properties. DNA chip technologies allow for the first time a global, simultaneous view of the transcription levels of many thousands of genes, under various cellular conditions. This opens great opportunities in medical, agricultural, and basic scientific research. A key step in the analysis of gene expression data is the identification of groups of genes that manifest similar expression patterns. This translates to the algorithmic problem of clustering gene expression data. The authors also discuss methods for evaluating the quality of clustering solutions in various situations, and demonstrate the performance of the algorithms on yeast cell cycle data.

The chapter by Kanehisa and Goto dsecribes the latest developments of the KEGG database. A key objective of the KEGG project is to computerize data and knowledge on molecular pathways and complexes that are involved in various cellular processes. Currently KEGG consists of (1) a pathway database, (2) a genes database, (3) a genome database, (4) a gene expression database, (5) a database of binary relations between proteins and other biological molecules, and (6) a ligand database, plus various classification information. It is well known that the analysis of individual molecules would not be sufficient for understanding higher order functions of cells and organisms. KEGG provides a computational resource for analyzing biological networks.

The chapter by Wong presents an introduction to what has come to be known as datamining and knowledge discovery in the biomedical context. The major reason that datamining has attracted increasing attention in the biomedical industry in recent years is due to the increased availability of huge amount of biomedical data and the imminent need to turn such data into useful information and knowledge. The knowledge gained can lead to improved drug targets, improved diagnostics, and improved treatment plans.

The last section of the book, which consists of six chapters, covers computational approaches for structure prediction and modeling of macromolecules.

Wang and Zhang's chapter presents an overview of predictions of RNA secondary structures. The secondary structure of an RNA is a set of base-pairs (nucleotide pairs) that form bonds between A-U and C-G. These bonds have been traditionally assumed to be noncrossing in a secondary structure. Two major prediction approaches considered are thermodynamic energy minimization methods and phylogenetic comparative methods. Thermodynamic energy minimization methods have been used to predict secondary structures from a single RNA sequence. Phylogenetic comparative methods have been used to determine secondary structures from a set of homologous RNAs whose sequences can be reliably aligned.

The chapter by Solovyev and Shindyalov provides a survey of computational methods for protein secondary structure predictions. Secondary structures describe regular features of the main chain of a protein molecule. Experimental investigation of polypeptides and small proteins suggest that a secondary structure can form in isolation, implying the possibility of identifying rules for its computational prediction. Predicting the secondary structure from an amino acid sequence alone is an important step toward our understanding of protein structures and functions. It may provide a starting point for tertiary structure modeling, especially in the absence of a suitable homologous template structure, reducing the search space in the simulation of protein folding.

The chapter by Chan et al. surveys currently available physics-based computational approaches to protein folding. A spectrum of methods—ranging from all-atom molecular dynamics to highly coarse-grained lattice modeling—have been employed to address physicochemical aspects of protein folding at various levels of structural and energetic resolution. The chapter discusses the strengths and limitations of some of these methods. In particular, the authors emphasize the primacy of self-contained chain models and how they differ logically from non-self-contained constructs with ad hoc conformational distributions. The important role of a protein's aqueous environment and the general non-additivity of solvent-mediated protein interactions are illustrated by examples in continuum electrostatics and atomic treatments of hydro-

phobic interactions. Several recent applications of simple lattice protein models are discussed in some detail.

In their chapter, Peitsch et al. discuss how protein models can be applied to functional analysis, as well as some of the current issues and limitations inherent to these methods. Functional analysis of the proteins discovered in fully sequenced genomes represents the next major challenge of life science research, and computational methods play an increasingly important part. Among them, comparative protein modeling will play a major role in this challenge, especially in light of the Structural Genomics programs about to be started around the world.

Xu and Xu's chapter presents a survey on protein threading as a computational technique for protein structure calculation. The fundamental reason for protein threading to be generally applicable is that the number of unique folds in nature is quite small, compared to the number of protein sequences, and a significant portion of these unique folds are already solved. A new trend in the development of computational modeling methods for protein structures, particularly in threading, is to incorporate partial structural information into the modeling process as constraints. This trend will become more clear as a great amount of structural data will be generated by the high-throughput structural genomics centers funded by the NIH Structural Genonics Initiative. The authors outline their recent work along this direction.

The chapter by Nussinov, Ma, and Wolson describes highly efficient, computer-vision and robotics based algorithms for docking and for the generation and matching of epitopes on molecular surfaces. The goal of frequently used approaches, both in searches for molecular similarity and for docking, that is, molecular complementarity, is to obtain highly accurate matching of respective molecular surfaces. Yet, owing to the variability of molecular surfaces in solution, to flexibility, to mutational events, and to the need to use modeled structures in addition to high resolution ones, utilization of epitopes may ultimately prove a more judicious approach to follow.

This book would not have been possible without the timely cooperation from all the authors and the patience of the publisher. Many friends and colleagues who have served as chapter reviewers have contributed tremendously to the quality and readability of the book. We would like to take this opportunity to thank them individually. They are: Nick Alexandrov, Vincent Berry, Mathieu Blanchette, David Bryant, Alberto Caprara, Kun-Mao Chao, Jean-Michel Claverie, Hui-Hsien Chou, Bhaskar DasGupta, Ramana Davuluri, Jim Fickett, Damian Gessler, Dan Gusfield, Loren Hauser, Xiaoqiu Huang, Larry Hunter, Shuyun Le, Sonia Leach, Hong Liu, Satoru Miyano, Ruth Nussinov, Victor Olman, Jose N. Onuchic, Larry Ruzzo, Gavin Sherlock, Jay Snoddy, Chao Tang, Ronald Taylor, John Tromp, Ilya A. Vakser, Martin Vingron, Natascha Vukasinovic, Mike Waterman, Liping Wei, Dong Xu, Zhenyu

Xuan, Lisa Yan, Louxin Zhang, and Zheng Zhang. We would also like to thank Ray Zhang for the artistic design of the cover page. Finally, we would like to thank Katherine Almeida, Katherine Innis, Ann Rae Jonas, Robert V. Prior, and Michael P. Rutter from The MIT Press for their great support and assistance throughout the process, and Dr. Guokui Liu for connecting us with the Tsinghua University Press (TUP) of China and facilitating copublication of this book by TUP in China.

	Preface	vii
I	**INTRODUCTION**	1
1	**The Challenges Facing Genomic Informatics** Temple F. Smith	3
II	**COMPARATIVE SEQUENCE AND GENOME ANALYSIS**	9
2	**Bayesian Modeling and Computation in Bioinformatics Research** Jun S. Liu	11
3	**Bio-Sequence Comparison and Applications** Xiaoqiu Huang	45
4	**Algorithmic Methods for Multiple Sequence Alignment** Tao Jiang and Lusheng Wang	71
5	**Phylogenetics and the Quartet Method** Paul Kearney	111
6	**Genome Rearrangement** David Sankoff and Nadia El-Mabrouk	135
7	**Compressing DNA Sequences** Ming Li	157
III	**DATA MINING AND PATTERN DISCOVERY**	173
8	**Linkage Analysis of Quantitative Traits** Shizhong Xu	175
9	**Finding Genes by Computer: Probabilistic and Discriminative Approaches** Victor V. Solovyev	201
10	**Computational Methods for Promoter Recognition** Michael Q. Zhang	249
11	**Algorithmic Approaches to Clustering Gene Expression Data** Ron Shamir and Roded Sharan	269
12	**KEGG for Computational Genomics** Minoru Kanehisa and Susumu Goto	301

13 **Datamining: Discovering Information from Bio-Data** 317
Limsoon Wong

IV **COMPUTATIONAL STRUCTURAL BIOLOGY** 343

14 **RNA Secondary Structure Prediction** 345
Zhuozhi Wang and Kaizhong Zhang

15 **Properties and Prediction of Protein Secondary Structure** 365
Victor V. Solovyev and Ilya N. Shindyalov

16 **Computational Methods for Protein Folding: Scaling a Hierarchy of Complexities** 403
Hue Sun Chan, Hüseyin Kaya, and Seishi Shimizu

17 **Protein Structure Prediction by Comparison: Homology-Based Modeling** 449
Manuel C. Peitsch, Torsten Schwede, Alexander Diemand, and Nicolas Guex

18 **Protein Structure Prediction by Protein Threading and Partial Experimental Data** 467
Ying Xu and Dong Xu

19 **Computational Methods for Docking and Applications to Drug Design: Functional Epitopes and Combinatorial Libraries** 503
Ruth Nussinov, Buyong Ma, and Haim J. Wolfson

Contributors 525
Index 527

I INTRODUCTION

1 The Challenges Facing Genomic Informatics

Temple F. Smith

What are these areas of intense research labeled bioinformatics and functional genomics? If we take literally much of the recently published "news and views," it seems that the often stated claim that the last century was the century of physics, whereas the twenty-first will be the century of biology, rests significantly on these new research areas. We might therefore ask: What is new about them? After all, computational or mathematical biology has been around for a long time. Surely much of bioinformatics, particularly that associated with evolution and genetic analyses, does not appear very new. In fact, the related work of researchers like R. A. Fisher, J. B. S. Haldane, and Sewell Wright dates nearly to the beginning of the 1900s. The modern analytical approaches to genetics, evolution, and ecology rest directly on their and similar work. Even genetic mapping easily dates to the 1930s, with the work of T. S. Painter and his students of *Drosophila* (still earlier if you include T. H. Morgan's work on X-linked markers in the fly). Thus a short historical review might provide a useful perspective on this anticipated century of biology and allow us to view the future from a firmer foundation.

First of all, it should be helpful to recognize that it was very early in the so-called century of physics that modern biology began, with a paper read by Hermann Müller at a 1921 meeting in Toronto. Müller, a student of Morgan's, stated that although of submicroscopic size, the gene was clearly a physical particle of complex structure, not just a working construct! Müller noted that the gene is unique from its product, and that it is normally duplicated unchanged, but once mutated, the new form is in turn duplicated faithfully.

The next 30 years, from the early 1920s to the early 1950s, were some of the most revolutionary in the science of biology. In my original field of physics, the great insights of relativity and quantum mechanics were already being taught to undergraduates; in biology, the new one-gene-one-enzyme concept was leading researchers to new understandings in biochemistry, genetics, and evolution. The detailed physical nature of the gene and its product were soon obtained. By midcentury, the unique linear nature of the protein and the gene were essentially known from the work of Frederick Sanger (Sanger 1949) and Erwin Chargraff (Chargraff 1950). All that remained was John Kendrew's structural analysis of sperm whale myoglobin (Kendrew 1958) and James Watson and Francis Crick's double helical model for DNA (Watson and Crick 1953). Thus by the mid-1950s, we had seen the physical gene and one of its products, and the motivation was in place to find them all. Of course, the genetic code needed to be determined and restriction enzymes discovered, but the beginning of modern molecular biology was on its way.

We might say that much of the last century was the century of applied physics, and the last half of the century was applied molecular biochemistry, generally called molecular biology! So what happened to create bioinformatics and functional genomics? It was, of course, the wealth of sequence data, first protein and then genomic. Both are based on some very clever chemistry and the late 1940s molecular sizing by chromatography. Frederick Sanger's sequencing of insulin (Sanger 1956) and Wally Gilbert and Allan Maxam's sequence of the Lactose operator from *E. coli* (Maxam and Gilbert 1977) showed that it could be done. Thus, in principle, all genetic sequences, including the human genome, were determinable; and, if determinable, they were surely able to be engineered, suggesting that the economics and even the ethics of biological research was about to change. The revolution was already visible to some by the 1970s.

The science or discipline of analyzing and organizing sequence data defines for many the bioinformatics realm. It had two somewhat independent beginnings. The older was the attempt to related amino acid sequences to the three-dimensional structure and function of proteins. The primary focus was the understanding of the sequence's encoding of structure and, in turn, the structure's encoding of biochemical function. Beginning with the early work of Sanger and Kendrew, progress continued such that, by the mid-1960s, Margaret Dayhoff (Dayhoff and Eck 1966) had formally created the first major database of protein sequences. By 1973, we had the start of the database of X-ray crystallographic determined protein atomic coordinates under Tom Koetzle at the Brookhaven National Laboratory.

From early on, Dayhoff seemed to understand that there was other very fundamental information available in sequence data, as shown in her many phylogenetic trees. This was articulated most clearly by Emile Zuckerkandl and Linus Pauling as early as 1965 (Zuckerkandl and Pauling 1965), that within the sequences lay their evolutionary history. There was a second fossil record to be deciphered.

It was that recognition that forms the true second beginning of what is so often thought of as the heart of bioinformatics, comparative sequence analyses. The seminal paper was by Walter Fitch and Emanuel Margoliash, in which they constructed a phylogenetic tree from a set of cytochrome sequences (Fitch and Margoliash 1967). With the advent of more formal analysis methods (Needleman and Wunsch 1970; Smith and Waterman 1981; Wilbur and Lipman 1983) and larger datasets (GenBank was started at Los Alamos in 1982), the marriage between sequence analysis and computer science emerged as naturally as it had with the analysis of tens of thousands of diffraction spots in protein structure determination a decade before. As if proof was needed that comparative sequence analysis was of more than academic interest, Russell Doolittle (Doolittle et al. 1983) demonstrated that we could explain the onc

gene v-sis's properties as an aberrant growth factor by assuming that related functions are carried out by sequence similar proteins.

By 1990, nearly all of the comparative sequence analysis methods had been refined and applied many times. The result was a wealth of new functional and evolutionary hypotheses. Many of these led directly to new insights and experimental validation. This in turn made the 40 years between 1950 and 1990 the years that brought reality to the dreams seeded in those wondrous previous 40 years of genetics and biochemistry. It is interesting to note that during this same 40 years, computers developed from the wartime monsters through the university mainframes and the lab bench workstation to the powerful personal computer. In fact, Doolittle's early successful comparative analysis was done on one of the first personal computers, an Apple II. The link between computers and molecular biology is further seen in the justification of initially placing GenBank at the Los Alamos National Laboratory rather than at an academic institution. This was due in large part to the laboratory's then immense computer resources, which in the year 2000 can be found in a top-of-the-line laptop!

What was new to computational biology was the data and the anticipated amount of it. Note that the human genome project was being formally initiated by 1990. Within the century's final decade, the genomes of more than two dozen microorganisms, along with yeast and *C. elegans*, the worm, would be completely sequenced. By the summer of the new century's very first year, the fruit fly genome would be sequenced, as well as 85 percent of the entire human genome. Although envisioned as possible by the late 1970s, no one foresaw the wealth of full genomic sequences that would be available at the start of the new millennium.

What challenges remained at the informatics level? Major database problems and some additional algorithm development will still surely come about. And, even though we still cannot predict a protein's structure or function directly from its sequence, *de novo*, straightforward sequence comparisons with such a wealth of data can generally infer both function and structure from the identification of close homologues previously analyzed. Yet it has slowly become obvious that there are at least four major problems here: first, most "previously analyzed" sequences obtained their annotation via sequence comparative inheritance, and not by any direct experimentation; second, many proteins carry out very different cellular roles even when their biochemical functions are similar; third, there are even proteins that have evolved to carry out functions distinct from those carried out by their close homologues (Jeffery 1999); and, finally, many proteins are multidomained and thus multifunctional, but identified by only one function. When we compound these facts with the lack of any universal vocabulary throughout much of molecular biology, there is great confusion, even with interpreting standard sequence similarity analysis. Even more to the point of the

future of bioinformatics is knowing that the function of a protein or even the role in the cell played by that function is only the starting point for asking real biological questions.

Asking questions beyond what biochemistry is encoded in a single protein or protein domain is still challenging. However, asking what role biochemistry plays in the life of the cell, which many refer to as functional genomics, is clearly even more challenging from the computational side. The analysis of genes and gene networks and their regulation may be even more complicated. Here we have to deal with alternate spliced gene products with potentially distinct functions and highly degenerate short DNA regulatory words. So far, sequence comparative methods have had limited success in these cases.

What will be the future role of computation in biology in the first few decades of this century? Surely many of the traditional comparative sequence analyses, including homologous extension protein structure modeling and DNA signal recognition, will continue to play major roles. As already demonstrated, standard statistical and clustering methods will be used on gene expression data. It is obvious, however, that the challenge for the biological sciences is to begin to understand how the genome parts list encodes cellular function—not the function of the individual parts, but that of the whole cell and organism. This, of course, has been the motivation underlying most of molecular biology over the last 20 years. The difference now is that we have the parts lists for multiple cellular organisms. These are complete parts lists rather than just a couple of genes identified by their mutational or other effects on a single pathway or cellular function. The past logic is now reversible: rather than starting with a pathway or physiological function, we can start with the parts list either to generate testable models or to carry out large-scale exploratory experimental tests. The latter, of course, is the logic behind the mRNA expression chips, whereas the former leads to experiments to test new regulatory network or metabolic pathway models. The design, analysis, and refinement of such complex models will surely require new computational approaches.

The analysis of the RNA expression data requires the identification of various correlations between individual gene expression profiles and between those profiles and different cellular environments or types. These, in turn, require some model concepts as to how the behavior of one gene may effect that of others, both temporally and spatially. Some straightforward analyses of RNA expression data have identified many differences in gene expression in cancer versus noncancer cells (Golub et al. 1999) and for different growth conditions (Eisen et al. 1998). Such data have also been used in an attempt to identify common or shared regulatory signals in bacteria (Hughes et al. 2000).

Yet expression data's full potential is not close to being realized. In particular, when gene expression data can be fully coupled to protein expression, modification, and activity, the very complex genetic networks should begin to come into view. In higher animals, for example, proteins can be complex products of genes through alternate exon splicing. We can anticipate that mRNA-based microarray expression analysis will be replaced by exon expression analysis. Here again, modeling will surely play a critical role, and the type of computational biology envisioned by population and evolutionary geneticists such as Wright may finally become a reality. This, the extraction of how the organism's range of behavior or environment responses is encoded in the genome, is the ultimate aim of functional genomics.

Many people in what is now called bioinformatics will recall that much of the wondrous mathematical modeling and analysis associated with population and evolutionary biology was at best suspect and at worst ignored by molecular biologists over the last 30 years or so. At the beginning of the new millennium, perhaps those thinkers should be viewed as being ahead of their time. Note, it was not that serious mathematics is not necessary to understand anything as complex as interacting populations, but only that the early biomodelers did not have the needed data! Today we are rapidly approaching the point where we can measure not only a population's genetic variation, but nearly all the genes that might be associated with a particular environmental response. It is the data that has created the latest aspect of the biological revolution. Just imagine what we will be able to do with a dataset composed of distributions of genetic variation among different subpopulations of fruit fly living in distinctly different environments, or what might we learn about our own evolution by having access to the full range of human and other primate genetic variation for all 40,000 to 100,000 human genes?

It is perhaps best for those anticipating the challenges of bioinformatics and computational genomics to think about how biology is likely to be taught by the end of the second decade of this century. Will the complex mammalian immune system be presented as a logical evolutionary adaptation of an early system for cell-cell communication that developed into a cell-cell recognition system, and then self-nonself recognition? Will it become obvious that the use by yeast of the G-protein couple receptors to recognize matting types would become one of the main components of nearly all higher organisms sensor systems? Like physics, where general rules and laws are taught at the start and the details are left for the computer, biology will surely be presented to future generations of students as a set of basic systems that have been duplicated and adapted to a very wide range of cellular and organismic functions following basic evolutionary principles constrained by Earth's geological history.

References

Chargraff, E. (1950). Chemical specificity of the nucleic acids and mechanisms of their enzymatic degradation. *Experimentia* 6: 201–208.

Dayhoff, M. O., and Eck, R. V. (1966). *Atlas of Protein Sequence and Structure*. Silver Spring, MD: NBRF Press.

Doolittle, R. F., Hunkapiller, M. W., Hood, L. E., Devare, S. G., Robbins, K. C., Aaronson, S. A., and Antoniades, H. N. (1983). Simian sarcoma virus onc gene, v-sis, is derived from the gene (or genes) encoding a platelet-derived growth factor. *Science* 221(4607): 275–277.

Eisen, M. B., Spellman, P. T., Brown, P. O., and Botstein, D. (1998). Cluster analysis and display of genome-wide expression patterns. *Proc. Natl. Acad. Sci. USA* 95(25): 14863–14868.

Fitch, W. M., and Margoliash, E. (1967). Construction of phylogenetic trees. A method based on mutation distances as estimated from cytochrome c sequences is of general applicability. *Science* 155: 279–284.

Golub, T. R., Slonim, D. K., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J. P., Coller, H., Loh, M. L., Downing, J. R., Caligiuri, M. A., Bloomfield, C. D., and Lander, E. S. (1999). Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science* 286(5439): 531–537.

Hughes, J. D., Estep, P. W., Tavazoie, S., and Church, G. M. (2000). Computational identification of cis-regulatory elements associated with groups of functionally related genes in Saccharomyces cerevisiae. *J. Mol. Biol.* 296(5): 1205–1214.

Jeffery, C. J. (1999). Moonlighting proteins. *Trends Biochem. Sci.* 24(1): 8–11.

Kendrew, J. C. (1958). The three-dimensional structure of a myoglobin. *Nature* 181: 662–666.

Maxam, A. M., and Gilbert, W. (1977). A new method for sequencing DNA. *Proc. Natl. Acad. Sci. USA* 74(2): 560–564.

Needleman, S. B., and Wunsch, C. D. (1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J. Mol. Biol.* 48: 443–453.

Sanger, F. (1949). *Cold Spring Harbor Symposia on Quantitative Biology* 14: 153–160.

Sanger, F. (1956). The structure of insulin. In *Currents in Biochemical Research*, Green, D. E. ed. New York: Interscience.

Smith, T. F., and Waterman, M. S. (1981). Identification of common molecular subsequences. *J. Mol. Biol.* 147: 195–197.

Watson, J. D., and Crick, F. H. C. (1953). Genetic implications of the structure of deoxyribonucleic acid. *Nature* 171: 964–967.

Wilbur, W. J., and Lipman, D. J. (1983). Rapid similarity searches of nucleic acid and protein data banks. *Proc. Natl. Acad. Sci. USA* 80(3): 726–730.

Zuckerkandl, E., and Pauling, L. C. (1965). Molecules as documents of evolutionary history. *J. Theoret. Biol.* 8: 357–358.

II COMPARATIVE SEQUENCE AND GENOME ANALYSIS

2 Bayesian Modeling and Computation in Bioinformatics Research

Jun S. Liu

2.1 Introduction

With the completion of decoding the human genome and genomes of many other species, the task of organizing and understanding the generated sequence and structural data becomes more and more pressing. These datasets also present great research opportunities to all quantitative researchers interested in biological problems. In the past decade, computational approaches to molecular and structural biology have attracted increasing attention from both laboratory biologists and mathematical scientists such as computer scientists, mathematicians, and statisticians, and have spawned the new field of bioinformatics. Among available computational methods, those that are developed based on explicit statistical models play an important role in the field and are the main focus of this chapter.

The use of probability theory and statistical principles in guarding against false optimism has been well understood by most scientists. The concepts of confidence interval, p-value, significance level, and the power of a statistical test routinely appear in scientific publications. To most scientists, these concepts represent, to a large extent, what statistics is about and what a statistician can contribute to a scientific problem. The invention of clever ideas, efficient algorithms, and general methodologies seem to be the privilege of scientific geniuses and are seldom attributed to a statistical methodology. In general, statistics or statistical thinking is not regarded as very helpful in attacking a difficult scientific problem. What we want to show here is that, quite in contrast to this "common wisdom," formal statistical modeling together with advanced statistical algorithms provide us a powerful "workbench" for developing innovative computational strategies and for making proper inferences to account for estimation uncertainties.

In the past decade, we have witnessed the developments of the likelihood approach to pairwise alignments (Bishop and Thompson 1986; Thorne et al. 1991); the probabilistic models for RNA secondary structure (Zuker 1989; Lowe and Eddy 1997); the expectation maximization (EM) algorithm for finding regulatory binding motifs (Lawrence and Reilly 1990; Cardon and Stormo 1992); the Gibbs sampling strategies for detecting subtle similarities (Lawrence et al. 1993; Liu 1994; Neuwald et al. 1997); the hidden Markov models (HMM) for DNA composition analysis and multiple alignments (Churchill 1989; Baldi et al. 1994; Krogh et al. 1994); and the hidden semi-Markov model for gene prediction and protein secondary structure prediction (Burge and Karlin 1997; Schmidler et al. 2000). All these developments show that algo-

rithms resulting from statistical modeling efforts constitute a major part of today's bioinformatics toolbox.

Our emphasis in this chapter is on the applications of the *Bayesian methodology* and its related algorithms in bioinformatics. We prefer a Bayesian approach for the following reasons: (1) its explicit use of probabilistic models to formulate scientific problems (i.e., a quantitative storytelling); (2) its coherent way of incorporating all sources of information and of treating *nuisance* parameters and missing data; and (3) its ability to quantify numerically uncertainties in all unknowns. In Bayesian analysis, a *comprehensive* probabilistic model is employed to describe relationships among various quantities under consideration: those that we observe (data and knowledge), those about which we wish to learn (scientific hypotheses), and those that are needed in order to construct a proper model (a scaffold). With this Bayesian model, the basic probability theory can automatically lead us to an efficient use of the available information when making predictions and to a numerical quantification of uncertainty in these predictions (Gelman et al. 1995). To date, statistical approaches have been primarily used in computational biology for deriving efficient algorithms. The utility of these methods to make statistical inferences about unobserved variables has received less attention.

An important yet subtle issue in applying the Bayes approach is the choice of a *prior* distribution for the unknown parameters. Because it is inevitable that we inject certain arbitrariness and subjective judgments into the analysis when prescribing a prior distribution, the Bayes methods have long been regarded as less "objective" than its frequentist counterpart (section 2.2), and thus, disfavored. Indeed, it is often nontrivial to choose an appropriate prior distribution when the parameter space is of a high dimension. All researchers who intend to use Bayesian methods for serious scientific studies need to put some thought into this issue. However, any scientific investigation has to involve a substantial amount of assumptions and personal judgements from the scientist(s) who conduct the investigation. These subjective elements, if made explicit and treated with care, should not undermine the scientific results of the investigation. More importantly, it should be regarded as a good scientific practice if the investigators make their subjective inputs explicit. Similarly, we argue that an appropriate subjective input in the form of a prior distribution should only enhance the relevance and accuracy of the Bayesian inference. Being able to make an explicit use of subjective knowledge is a virtue, instead of blemish, of Bayesian methods.

This chapter is organized as follows. Section 2.2 discusses the importance of formal statistical modeling and gives an overview of two main approaches to statistical inference: the frequentist and Bayesian. Section 2.3 outlines the Bayesian procedure

for treating a statistical problem, with an emphasis on using the missing data formulation to construct scientifically meaningful models. Section 2.4 describes several popular algorithms for dealing with statistical computations: the EM algorithm, the Metropolis algorithm, and the Gibbs sampler. Section 2.5 demonstrates how the Bayesian method can be used to study a sequence composition problem. Section 2.6 gives a further example of using the Bayesian method to find subtle repetitive motifs in a DNA sequence. Section 2.7 concludes the chapter with a brief discussion.

2.2 Statistical Modeling and Inference

2.2.1 Parametric Statistical Modeling

Statistical modeling and analysis, including the collection of data, the construction of a probabilistic model, the quantification and incorporation of expert opinions, the interpretation of the model and the results, and the prediction from the data, form an essential part of the scientific method in diverse fields. The key focus of statistics is on making inferences, where the word *inference* follows the dictionary definition as "the process of deriving a conclusion from fact and/or premise." In statistics, the facts are the observed data, the premise is represented by a probabilistic model of the system of interest, and the conclusions concern unobserved quantities. Statistical inference distinguishes itself from other forms of inferences by explicitly quantifying uncertainties involved in the premise and the conclusions.

In *nonparametric* statistical inference, one does not assume any specific distributional form for the probability law of the observed data, but only imposes on the data a dependence (or independence) structure. For example, an often imposed assumption in nonparametric analyses is that the observations are *independent and identically distributed* (iid). When the observed data are continuous quantities, what one has to infer for this nonparametric model is the whole density curve—an infinite dimensional parameter. A main advantage of nonparametric methods is that the resulting inferential statements are relatively more robust than those from parametric methods. However, a main disadvantage of the nonparametric approach is that it is difficult, and sometimes impossible, to build into the model more sophisticated structures (based on our scientific knowledge). It does not facilitate "learning."

Indeed, it would be ideal and preferable if we could derive what we want without having to assume anything. However, the process of using simple models (with a small number of adjustable parameters) to describe natural phenomena and then improving upon them (e.g., Newton's law of motion versus Einstein's theory of relativity) is at the heart of all scientific investigations. Parametric modeling, either analytically or qualitatively, either explicitly or implicitly, is intrinsic to human intelligence; it is the

only way we learn about the outside world. Analogously, statistical analysis based on parametric modeling is also essential to our scientific understanding of the data.

At a conceptual level, probabilistic models in statistical analyses serve as a mechanism through which one connects observed data with a scientific premise or hypothesis about real-world phenomena. Because bioinformatics explicitly or implicitly concerns the analysis of biological data that are intrinsically probabilistic, such models should be also at the core of bioinformatics. No model can completely represent every detail of reality. The goal of modeling is to abstract the key features of the underlying scientific problem into a workable mathematical form with which the scientific premise may be examined. Families of probability distributions characterized by a small number of parameters are most useful for this purpose.

Let $\mathbf{y}$ denote the observed data. In *parametric inference*, we assume that the observation follows a probabilistic law that belongs to a given distribution family. That is, $\mathbf{y}$ is a realization of a random process (i.e., a sample from a distribution) whose probability law has a particular form (e.g., Gaussian, multinomial, Dirichlet, etc.), $f(\mathbf{y} \mid \boldsymbol{\theta})$, which is completely known other than $\boldsymbol{\theta}$. Here $\boldsymbol{\theta}$ is called a (population) *parameter*, and it often corresponds to a scientific premise for our understanding of a natural process. To be concrete, one can imagine that $\mathbf{y}$ is a genomic segment of length n from a certain species, say, human. The simplest probabilistic model for a genomic segment is the "iid model," in which every observed DNA base pair (bp) in the segment is regarded as *independent* of others and produced randomly by nature based on a roll of a four-sided die (maybe loaded). Although very simple and unrealistic, this model is the so-called "null model" behind almost all theoretical analyses of popular biocomputing methods. That is, if we want to assess whether a pattern we find can be regarded as a "surprise," the most natural analysis is to evaluate how likely this pattern will occur if an iid model is assumed.

Finding a value of $\boldsymbol{\theta}$ that is most compatible with the observation $\mathbf{y}$ is termed as *model fitting* or *estimation*. We make scientific progresses by iterating between fitting the data to the posited model and proposing an improved model to accommodate important features of the data that are not accounted for by the previous model. When the model is given, an efficient method should be used to make inference on the parameters. Both the maximum likelihood estimation method and the Bayes method use the likelihood function to extract information from data and are efficient; these methods will be the main focus of the remaining part of this chapter.

2.2.2 Frequentist Approach to Statistical Inference

The frequentist approach, sometimes simply referred to as the classical statistics procedure, arrives at its inferential statements by using a point estimate of the un-

known parameter and addressing the estimation uncertainty by the *frequency behavior* of the estimator. Among all estimation methods, the method of *maximum likelihood estimate* (MLE) is most popular.

The MLE of $\boldsymbol{\theta}$ is defined as an argument $\hat{\boldsymbol{\theta}}$ that maximizes the likelihood function, that is,

$$\hat{\boldsymbol{\theta}} = \arg \max_{\text{all } \boldsymbol{\theta}} L(\boldsymbol{\theta} \mid \mathbf{y})$$

where the *likelihood function* $L(\boldsymbol{\theta} \mid \mathbf{y})$ is defined to be *any* function that is proportional to the probability density $f(\mathbf{y} \mid \boldsymbol{\theta})$. Clearly, $\hat{\boldsymbol{\theta}}$ is a function of $\mathbf{y}$ and its form is determined completely by the parametric model $f(\ )$. Hence, we can write $\hat{\boldsymbol{\theta}}$ as $\hat{\boldsymbol{\theta}}(\mathbf{y})$ to explicate this connection. Any deterministic function of the data $\mathbf{y}$, such as $\hat{\boldsymbol{\theta}}(\mathbf{y})$, is called an *estimator*. For example, if $\mathbf{y} = (y_1, \ldots, y_n)$ are iid observations from $N(\theta, 1)$, a Normal distribution with mean θ and variance 1, then the MLE of θ is $\hat{\theta}(\mathbf{y}) = \bar{y}$, the sample mean of the y_i, which is a linear combination of the y. It can be shown that, under regularity conditions, the MLE $\hat{\boldsymbol{\theta}}(\mathbf{y})$ is asymptotically most efficient among all potential estimators. In other words, no other way of using $\mathbf{y}$ can perform better asymptotically, in terms of estimating $\boldsymbol{\theta}$, than the MLE procedure. But some inferior methods, such as the method of moments (MOM), can be used as alternatives when the MLE is difficult to obtain.

Uncertainty in estimation is addressed by the *principle of repeated sampling*. Imagine that the same stochastic process that "generates" our observation $\mathbf{y}$ can be repeated indefinitely under identical conditions. A frequentist studies what the "typical" behavior of an estimator, for example, $\hat{\theta}(\mathbf{y}_{\text{rep}})$, is. Here $\mathbf{y}_{\text{rep}}$ denotes a hypothetical dataset generated by a replication of the *same* process that generates y and is, therefore, a random variable that has $\mathbf{y}$'s characteristics. The distribution of $\hat{\theta}(\mathbf{y}_{\text{rep}})$ is called the *frequency behavior* of estimator $\hat{\theta}$. For the Normal example, the frequency distribution of $\bar{y}_{\text{rep}}$ is $N(\theta, 1/n)$. With this distribution available, we can calibrate the observed $\hat{\theta}(\mathbf{y})$ with the "typical" behavior of $\hat{\theta}(\mathbf{y}_{\text{rep}})$, such as $N(\theta, 1/n)$, to quantify uncertainty in the estimation. As another example, suppose $\mathbf{y} = (y_1, \ldots, y_n)$ is a genomic segment and let n_a be the number of "A"s in $\mathbf{y}$. Then $\hat{\theta}_a = n_a/n$ is an estimator of θ_a, the "true frequency of A" under the iid die-rolling model. To understand the uncertainty in $\hat{\theta}_a$, we need to go back to the iid model and ask ourselves: How would n_a fluctuate in a segment like $\mathbf{y}$ that is generated by the same die-rolling process? The answer is rather simple: n_a follows distribution $\text{Binom}(n, \theta_a)$ and has mean $n\theta_a$ and variance $n\theta_a(1 - \theta_a)$.

We want to emphasize that the concepts of an "estimator" and its uncertainty only make sense if a generative model is contemplated. For example, the statement that

"$\hat{\theta}_a$ estimates the true frequency of A" only makes sense if we imagine that an iid model (or another similar model) was used to generate the data. If this model is not really what we have in mind, then the meaning of $\hat{\theta}_a$ is no longer clear. A imaginary random process for the data generation is crucial for deriving a valid statistical statement.

A $(1-\alpha)100\%$ confidence interval (or region) for $\boldsymbol{\theta}$, for instance, is of the form $(\underline{\boldsymbol{\theta}}(\mathbf{y}_{\text{rep}}), \bar{\boldsymbol{\theta}}(\mathbf{y}_{\text{rep}}))$, meaning that under repeated sampling, the probability that the interval (the interval is random under repeated sampling) covers the true θ is at least $1-\alpha$. In contrast to what most people have hoped for, this interval statement *does not* mean that "$\boldsymbol{\theta}$ is in $(\underline{\boldsymbol{\theta}}(\mathbf{y}), \bar{\boldsymbol{\theta}}(\mathbf{y}))$ with probability $1-\alpha$." With observed $\mathbf{y}$, the true $\boldsymbol{\theta}$ is either in or out of the interval and no meaningful direct probability statement can be given unless $\boldsymbol{\theta}$ can be treated as a random variable.

When finding the analytical form of the frequency distribution of an estimator $\hat{\boldsymbol{\theta}}$ is difficult, some modern techniques such as the *jackknife* or *bootstrap* method can be applied to numerically simulate the "typical" behavior of an estimator (Efron 1979). Suppose $\mathbf{y} = (y_1, \ldots, y_n)$ and each y_i follows an iid model. In the bootstrap method, one treats the empirical distribution of $\mathbf{y}$ (the distribution that gives a probability mass of $1/n$ to each y_i and 0 to all other points in the space) as the "true underlying distribution" and repeatedly generates new datasets, $\mathbf{y}_{\text{rep},1}, \ldots, \mathbf{y}_{\text{rep},B}$, from this distribution. Operationally, each $\mathbf{y}_{\text{rep},b}$ consists of n data points, $\mathbf{y}_{\text{rep},b} = (y_{b,1}, \ldots, y_{b,n})$, where each $y_{b,i}$ is a simple random sample (with replacement) from the set of the observed data points $\{y_1, \ldots, y_n\}$. With the bootstrap samples, we can calculate $\hat{\theta}(\mathbf{y}_{\text{rep},b})$ for $b = 1, \ldots, B$, whose histogram tells us how $\hat{\theta}$ varies from sample to sample assuming that the true distribution of $\mathbf{y}$ is its observed empirical distribution.

In a sense, the classical inferential statements are *pre-data* statements because they are concerned with the repeated sampling properties of a procedure and do not have to refer to the actual observed data (except in the bootstrap method, where the observed data is used in the approximation of the "true underlying distribution"). A major difficulty in the frequentist approach, besides its awkwardness in quantifying estimation uncertainty, is its difficulty in dealing with nuisance parameters. Suppose $\boldsymbol{\theta} = (\theta_1, \theta_2)$. In a problem where we are only interested in one component, θ_1 say, the other component θ_2 becomes a *nuisance parameter*. No clear principles exist in classical statistics that enable us to eliminate θ_2 in an optimal way. One of the most popular practices in statistical analysis is the so-called *profile likelihood* method, in which one treats the nuisance parameter θ_2 as known and fixes it at its MLE. This method, however, underestimates the involved uncertainty (because it treats unknown θ_2 as if it were known) and can lead to incorrect inference when the distribution of $\hat{\theta}_1$ depends on θ_2, especially if the dimensionality of θ_2 is high. More sophisticated

methods based on orthogonality, similarity, and average likelihood have also been proposed, but they all have their own problems and limitations.

2.2.3 Bayesian Methodology

Bayesian statistics seeks a more ambitious goal by modeling all related information and uncertainty, such as physical randomness, subjective opinions, prior knowledge from different sources, and so on, with a joint probability distribution and treating *all quantities* involved in the model, be they observations, missing data, or unknown parameters, as random variables. It uses the calculus of probability as the guiding principle in manipulating data and derives its inferential statements based purely on an appropriate conditional distribution of unknown variables.

Instead of treating $\boldsymbol{\theta}$ as an unknown constant as in a frequentist approach, Bayesian analysis treats $\boldsymbol{\theta}$ as a realized value of a random variable that follows a *prior distribution* $f_0(\boldsymbol{\theta})$, which is typically regarded as known to the researcher independently of the data under analysis. The Bayesian approach has at least two advantages. First, through the prior distribution, we can inject prior knowledge and information about the value of $\boldsymbol{\theta}$. This is especially important in bioinformatics, as biologists often have substantial knowledge about the subject under study. To the extent that this information is correct, it will sharpen the inference about $\boldsymbol{\theta}$. Second, treating all the variables in the system as random variables greatly clarifies the methods of analysis. It follows from the basic probability theory that information about the realized value of any random variable, $\boldsymbol{\theta}$, say, based on observation of related random variables, $\mathbf{y}$, say, is summarized in the conditional distribution of $\boldsymbol{\theta}$ given $\mathbf{y}$, the so-called *posterior distribution*. Hence, if we are interested only in a component of $\boldsymbol{\theta} = (\theta_1, \theta_2)$, say θ_1, we have just to integrate out the remaining components of $\boldsymbol{\theta}$, the nuisance parameters, from the posterior distribution. Furthermore, if we are interested in the prediction of a future observation $\mathbf{y}^+$ depending on $\boldsymbol{\theta}$, we can obtain the posterior distribution of $\mathbf{y}^+$ given $\mathbf{y}$ by completely integrating out $\boldsymbol{\theta}$.

The use of probability distributions to describe unknown quantities is also supported by the fact that probability theory is the only known coherent system for quantifying objective and subjective uncertainties. Furthermore, probabilistic models have been accepted as appropriate in almost all information-based technologies, including information theory, control theory, system science, communication and signal processing, and statistics. When the system under study is modeled properly, the Bayesian approach is coherent, consistent, and efficient.

The theorem that combines the prior and the data to form the posterior distribution (section 2.3) is a simple mathematical result first given by Thomas Bayes in 1763. The statistical procedure based on the systematic use of this theorem appears much

later (some people believe that Laplace was the first *Bayesian*) and is also named after Bayes. The adjective *Bayesian* is often used for approaches in which subjective probabilities are emphasized. In this sense, Thomas Bayes was not really a Bayesian.

A main controversial aspect of the Bayesian approach is the use of the prior distribution, to which three interpretations can be given: (1) as frequency distributions; (2) as objective representations of a rational belief of the parameter, usually in a state of ignorance; and (3) as a subjective measure of what a particular individual believes (Cox and Hinkley 1974). Interpretation (1) refers to the case when $\boldsymbol{\theta}$ indeed follows a stochastic process and, therefore, is uncontroversial. But this scenario is of limited applicability. Interpretation (2) is theoretically interesting but is often untenable in real applications. The emotive words "subjective" and "objective" should not be taken too seriously. (Many people regard the frequentist approach as a more "objective" one.) There are considerable subjective elements and personal judgements injected into all phases of scientific investigations. Claiming that someone's procedure is "more objective" based on how the procedure is derived is nearly meaningless. A truly objective evaluation of any procedure is how well it attains its stated goals. In bioinformatics, we are fortunate to have a lot of known biological facts to serve as objective judges.

In most of our applications, we employ the Bayesian method mainly because of its internal consistency in modeling and analysis and its capability to combine various sources of information. Thus, we often take a combination of (1) and (3) for deriving a "reasonable" prior for our data analysis. We advocate the use of a suitable sensitivity analysis, that is, an analysis of how our inferential statements are influenced by a change in the prior, to validate our statistical conclusions.

2.2.4 Connection with Some Methods in Bioinformatics

Nearly all bioinformatics methods employ score functions—which are often functions of likelihoods or likelihood ratios—at least implicitly. The specification of priors required for Bayesian statistics is less familiar in bioinformatics, although not completely foreign. For example, the setting of parameters for an alignment algorithm can be viewed as a special case of prior specification in which the prior distribution is degenerate with probability one for the set value and zero for all other values. The introduction of non-degenerate priors can typically give more flexibility in modeling reality.

The use of formal statistical models in bioinformatics was relatively rare before the 1990s. One reason is perhaps that computer scientists, statisticians, and other data analysts were not comfortable with big models—it is hard to think about many unknowns simultaneously. Additionally, algorithms for dealing with complex statis-

tical models were not sufficiently well known and the computer hardware was not yet as powerful. Recently, an extensive use of probabilistic models (e.g., the hidden Markov model and the missing data formalism) has contributed greatly to the advance of computational biology.

Recursive algorithms for global optimization have been employed with great advantage in bioinformatics as the basis of a number of dynamic programming algorithms. We show that these algorithms have very similar counterparts in Bayesian and likelihood computations.

2.3 Bayes Procedure

2.3.1 The Joint and Posterior Distributions

The full process of a typical Bayesian analysis can be described as consisting of three main steps (Gelman et al. 1995): (1) setting up a full probability model, the *joint distribution*, that captures the relationship among *all* the variables (e.g., observed data, missing data, unknown parameters) in consideration; (2) summarizing the findings for particular quantities of interest by appropriate posterior distributions, which is typically a conditional distribution of the quantities of interest given the observed data; and (3) evaluating the appropriateness of the model and suggesting improvements (model criticism and selection).

A standard procedure for carrying out step (1) is to formulate the scientific question of interest though the use of a probabilistic model, from which we can write down the *likelihood function* of $\boldsymbol{\theta}$. Then a prior distribution $f_0(\boldsymbol{\theta})$ is contemplated, which should be both mathematically tractable and scientifically meaningful. The joint probability distribution can then be represented as *Joint* = *likelihood* × *prior*, that is,

$$p(\mathbf{y}, \boldsymbol{\theta}) = p(\mathbf{y} \mid \boldsymbol{\theta}) f_0(\boldsymbol{\theta}) \tag{2.1}$$

For notational simplicity, we use $p(\mathbf{y} \mid \boldsymbol{\theta})$, hereafter, interchangeably with $f(\mathbf{y} \mid \boldsymbol{\theta})$ to denote the likelihood. From a Bayesian's point of view, this is simply a conditional distribution.

Step (2) is completed by obtaining the *posterior distribution* through the application of Bayes theorem:

$$p(\boldsymbol{\theta} \mid \mathbf{y}) = \frac{p(\mathbf{y}, \boldsymbol{\theta})}{p(\mathbf{y})} = \frac{p(\mathbf{y} \mid \boldsymbol{\theta}) f_0(\boldsymbol{\theta})}{\int p(\mathbf{y} \mid \boldsymbol{\theta}) f_0(\boldsymbol{\theta})\, d\boldsymbol{\theta}} \propto p(\mathbf{y} \mid \boldsymbol{\theta}) f_0(\boldsymbol{\theta}) \tag{2.2}$$

When $\boldsymbol{\theta}$ is discrete, the integral is replaced by summation. The denominator $p(\mathbf{y})$, which is a normalizing constant for the function, is sometimes called the *marginal*

likelihood of the model and can be used to conduct model selection (Kass and Raftery 1995). Although evaluating $p(\mathbf{y})$ analytically is infeasible in many applications, Markov chain Monte Carlo methods (section 2.4) can often be employed for its estimation.

In computational biology, because the data to be analyzed are usually categorical (e.g., DNA sequences with a four-letter alphabet or protein sequences with a twenty-letter alphabet), the *multinomial distribution* is most commonly used. The parameter vector $\boldsymbol{\theta}$ in this model corresponds to the frequencies of each base type in the data. A mathematically convenient prior distribution for the multinomial families is the *Dirichlet distributions*, of which the Beta distribution is a special case for the binomial family. This distribution has the form

$$f_0(\boldsymbol{\theta}) \propto \prod_{j=1}^{k} \theta_j^{\alpha_j - 1} \tag{2.3}$$

where k is the size of the alphabet and $\alpha_j > 0$ for all j. Here $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_k)$ is often called the hyper-parameter for the Dirichlet distribution and the sum $\alpha = \alpha_1 + \cdots + \alpha_k$ is often called the "pseudo-counts," which can be understood heuristically as the total "worth" (in comparison with actual observations) of one's prior opinion. When a simple iid model is imposed on an observed sequence of letters, $\mathbf{y} = (y_1, \ldots, y_n)$, its likelihood function is

$$p(\mathbf{y} \mid \boldsymbol{\theta}) = \prod_{i=1}^{n} \theta_{y_i} = \prod_{j=1}^{k} \theta_j^{n_j}$$

where n_j is the number of counts of residual type j in $\mathbf{y}$. If a Dirichlet($\boldsymbol{\alpha}$) prior used for its parameter $\boldsymbol{\theta}$, the posterior distribution for $\boldsymbol{\theta}$ is simply another Dirichlet distribution with hyperparameter $(\alpha_1 + n_1, \ldots, \alpha_k + n_k)$. The posterior mean of, say, θ_j, is $(n_j + \alpha_j)/(n + \alpha)$.

Suppose the parameter vector has more than one component, that is, $\boldsymbol{\theta} = (\theta_1, \boldsymbol{\theta}_{[-1]})$, where $\boldsymbol{\theta}_{[-1]}$ denotes all but the first component. One may be interested only in one of components, θ_1, say. The other components that are not of immediate interest but are needed by the model, *nuisance parameters*, can be removed by integration:

$$\begin{aligned} p(\theta_1 \mid \mathbf{y}) &= \frac{p(\mathbf{y}, \theta_1)}{p(\mathbf{y})} \\ &= \frac{\int p(\mathbf{y} \mid \theta_1, \boldsymbol{\theta}_{[-1]}) f_0(\theta_1, \boldsymbol{\theta}_{[-1]})\, d\boldsymbol{\theta}_{[-1]}}{\int\int p(\mathbf{y} \mid \theta_1, \boldsymbol{\theta}_{[-1]}) f_0(\theta_1, \boldsymbol{\theta}_{[-1]})\, d\theta_1\, d\boldsymbol{\theta}_{[-1]}} \end{aligned} \tag{2.4}$$

Note that computations required for completing a Bayesian inference are the integrations (or summations for discrete parameters) over all unknowns in the joint distribution to obtain the marginal likelihood and over all but those of interest to remove nuisance parameters. Despite the deceptively simple-looking form of equation (2.4), the challenging aspects of Bayesian statistics are: (1) the development of a model, $p(\mathbf{y} \mid \boldsymbol{\theta}) f_0(\boldsymbol{\theta})$, which must effectively capture the key features of the underlying scientific problem; and (2) the necessary computation for deriving the posterior distribution. For aspect (1), the *missing data* formulation is an important tool to help one formulate a scientific problem; for (2), the recent advances in Markov chain Monte Carlo techniques are essential.

2.3.2 The Missing Data Framework

The missing data formulation is an important methodology for modeling complex data structures and for designing computational strategies. This general framework was motivated in early 1970s (and maybe earlier) by the need for a proper statistical analysis of certain survey data where parts of the data were missing. For example, a large survey of families was conducted in 1967 in which many socioeconomic variables were recorded. A follow-up study of the same families was done in 1970. Naturally, the 1967 data had a large amount of missing values due to either recording errors or some families' refusal to answer certain questions. The 1970 data had an even more severe kind of missing data caused by the fact that many families studied in 1967 could not be located in 1970.

The first important question for a missing data problem is under what conditions can we ignore the "missing mechanism" in the analysis. That is, does the fact that an observation is missing tell us anything about the quantities we are interested in estimating? For example, the fact that many families moved out of a particular region may indicate that the region's economy was having problems. Thus, if our interested estimand is a certain "consumer confidence" measure of the region, the standard estimate resulting only from the observed families might be biased. Rubin's (1976) pioneering work provides general guidance on how to judge the ignorability. Because everything in a Bayes model is a random variable, it is especially convenient and transparent in dealing with these ignorability problems in a Bayesian framework. The second important question is that how one should conduct computations, such as finding the MLE or the posterior distribution of the estimands. This question has motivated statisticians to develop several important algorithms: the EM algorithm (Dempster et al. 1977), data augmentation (Tanner and Wong 1987), and the Gibbs sampler (Gelfand and Smith 1990).

In late 1970s and early 1980s, people started to realize that many other problems can be treated as missing data problems. One typical example is the so-called *latent-class* model, which is most easily explained by the following example (Tanner and Wong 1987). In the 1972–1974 General Social Surveys, a sample of 3,181 participants were asked to answer the following questions. Question A: Do you think it should be possible for a pregnant woman to obtain a legal abortion *if she is married and does not want any more children.* In question B, the italicized phrase in A is replaced with "if she is not married and does not want to marry the man." A latent-class model assumes that a person's answers to A and B are conditionally independent given the value of a dichotomous latent variable Z (either 0 or 1). Intuitively, this model asserts that the population consists of two "types" of people (e.g., conservative and liberal) and Z is the unobserved "label" of each person. If you know the person's label, then his/her answer to question A will not help you to predict his/her answer to question B. Clearly, variable Z can be thought of as a "missing data," although it is not really "missing" in a standard sense. For another example, in a multiple sequence alignment problem, alignment variables that must be specified for each sequence (observation) can be regarded as missing data. Residue frequencies or scoring matrices, which apply to all the sequences, are population parameters. This generalized view eventually made the missing data formulation one of the most versatile and constructive workbenches for sophisticated statistical analysis and advanced statistical computing.

The importance of the missing data formulation stems from the following two main considerations. Conceptually, this framework helps in making model assumptions explicit (e.g., ignorable versus nonignorable missing mechanism), in defining precise estimands of interest, and in providing a logical framework for causal inference (Rubin 1976). Computationally, the missing data formulation inspired the invention of several important statistical algorithms. Mathematically, however, the missing data formulation is not well defined. In real life, what we can observe is always partial (incomplete) information and there is no absolute distinction between parameters and missing data (i.e., some unknown parameters can also be thought of as missing data, and vice versa).

To a broader scientific audience, the concept of "missing data" is perhaps a little odd because many scientists may not believe that they have any missing data. In the most general and abstract form, the "missing data" can refer to any unobserved component of the probabilistic system under consideration and the inclusion of this part in the system often results in a simpler structure. This component, however, needs to be marginalized (integrated) out in the final analysis. That is, when missing data $\mathbf{y}_{\text{mis}}$ is present, a proper inference about the parameters of interest can be achieved

by using the "observed-data likelihood," $L_{\text{obs}}(\theta; \mathbf{y}_{\text{obs}}) = p(\mathbf{y}_{\text{obs}} \mid \boldsymbol{\theta})$, which can be obtained by integration:

$$L_{\text{obs}}(\boldsymbol{\theta}; \mathbf{y}_{\text{obs}}) \propto \int p(\mathbf{y}_{\text{obs}}, \mathbf{y}_{\text{mis}} \mid \boldsymbol{\theta})\, d\mathbf{y}_{\text{mis}}$$

Because it is often difficult to compute this integral analytically, one needs advanced computational methods such as the EM algorithm (Dempster et al. 1977) to compute the MLE.

Bayesian analysis for missing data problems can be achieved coherently through integration. Let $\boldsymbol{\theta} = (\theta_1, \boldsymbol{\theta}_{[-1]})$ and suppose we are interested only in θ_1. Then

$$p(\theta_1 \mid \mathbf{y}_{\text{obs}}) \propto \iint p(\mathbf{y}_{\text{obs}}, \mathbf{y}_{\text{mis}} \mid \theta_1, \boldsymbol{\theta}_{[-1]}) p(\theta_1, \boldsymbol{\theta}_{[-1]})\, d\mathbf{y}_{\text{mis}}\, d\boldsymbol{\theta}_{[-1]}$$

Because all quantities in a Bayesian model are treated as random variables, the integration for eliminating the missing data is no different than that for eliminating nuisance parameters.

Our main use of the missing data formulation is to construct proper statistical models for bioinformatics problems. As will be shown in the later sections, this framework frees us from being afraid of introducing meaningful but perhaps high-dimensional variables into our model, which is often necessary for a satisfactory description of the underlying scientific knowledge. The extra variables introduced this way, when treated as missing data, can be integrated out in the analysis stage so as to result in a proper inference for the parameter of interest. Although a conceptually simple procedure, the computation involved in integrating out missing data can be very difficult. Section 2.4 introduces a few algorithms for this purpose.

2.3.3 Model Selection and Bayes Evidence

At times, biology may indicate that more than one model is plausible. Then we are interested in assessing model fit and conducting model selection (step [3] described in section 2.2.1). Classical hypothesis testing can be seen as a model selection method in which one chooses between the null hypothesis and the alternative in light of data. Model selection can also be achieved by a formal Bayes procedure. First, all the candidate models are embedded into one unified model. Then the "overall" posterior probability of each candidate model is computed and used to discriminate among the models (Kass and Raftery 1995).

To illustrate the Bayes procedure for model selection, we focus on the comparison of two models: $M = 0$ indicates the "null" model, and $M = 1$ the alternative. The joint distribution for the *augmented model* becomes

$$p(\mathbf{y}, \boldsymbol{\theta}, M) = p(\mathbf{y} \mid \boldsymbol{\theta}, M) p(\boldsymbol{\theta}, M)$$

Under the assumption that the data depend on the models through their respective parameters, the above equation is equal to

$$p(\mathbf{y}, \boldsymbol{\theta}, M) = p(\mathbf{y} \mid \boldsymbol{\theta}_m) p(\boldsymbol{\theta}_m \mid M = m) p(M = m)$$

where $p(\boldsymbol{\theta}_m \mid M = m)$ is the prior for the parameters in model m, and $p(M = m)$ is the prior probability of model m. Note that the dimensionality of $\boldsymbol{\theta}_m$ may be different for different m. The posterior probability for model m is obtained as:

$$\begin{aligned} p(M = m \mid \mathbf{y}) &\propto p(\mathbf{y} \mid M = m) p(M = m) \\ &= \left\{ \int p(\mathbf{y} \mid \boldsymbol{\theta}_m) p(\boldsymbol{\theta}_m \mid M = m) \, d\boldsymbol{\theta}_m \right\} p(M = m) \end{aligned}$$

The choice of $p(M = m)$, which is our prior on different models, is assigned independently of data in study. A frequent choice is $p(M = 0) = p(M = 1) = 0.5$ if we expect that both models are equally likely a priori. But in other cases, we might set $p(M = 1)$ very small. For example, in database searching, the prior probability that the query sequence is related to a sequence taken at random from the database is much smaller. In this case we might set $p(M = 1)$ inversely proportional to the number of sequences in the database.

2.4 Advanced Computation in Statistical Analysis

In many practical problems, the required computation is the main obstacle for applying both the Bayesian and the MLE methods. In fact, until recently, these computations have often been so difficult that sophisticated statistical modeling and Bayesian methods were largely for theoreticians and philosophers. The introduction of the bootstrap method (Efron 1979), the expectation maximization (EM) algorithm (Dempster et al. 1977), and the Markov chain Monte Carlo (MCMC) method (Gilks et al. 1998) has brought many powerful statistical models into the mainstream of statistical analysis. As we illustrate in section 2.5, by appealing to the rich history of computation in bioinformatics, many required optimizations and integrations can be done exactly, which gives rise to either an exact solution to the MLE and the posterior distributions or an improved MCMC algorithm.

2.4.1 The EM Algorithm

The EM algorithm is perhaps one of the most well-known statistical algorithms for finding the mode of a *marginal* likelihood or posterior distribution function. That is,

the EM algorithm enables one to find the mode of

$$F(\boldsymbol{\theta}) = \int f(\mathbf{y}_{\text{mis}}, \mathbf{y}_{\text{obs}} \mid \boldsymbol{\theta})\, d\mathbf{y}_{\text{mis}} \tag{2.5}$$

where $f(\mathbf{y}_{\text{mis}}, \mathbf{y}_{\text{obs}} \mid \boldsymbol{\theta}) \geq 0$ and $F(\boldsymbol{\theta}) < \infty$ for all $\boldsymbol{\theta}$. When $\mathbf{y}_{\text{mis}}$ is discrete, we simply replace the integral in equation (2.5) by summation. The EM algorithm starts with an initial guess $\boldsymbol{\theta}^{(0)}$ and iterates the following two steps:

- **E-step.** Compute

$$\begin{aligned} Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^{(t)}) &= E_t[\log f(\mathbf{y}_{\text{mis}}, \mathbf{y}_{\text{obs}} \mid \boldsymbol{\theta}) \mid \mathbf{y}_{\text{obs}}] \\ &= \int \log f(\mathbf{y}_{\text{mis}}, \mathbf{y}_{\text{obs}} \mid \boldsymbol{\theta}) f(\mathbf{y}_{\text{mis}} \mid \mathbf{y}_{\text{obs}}, \boldsymbol{\theta}^{(t)})\, d\mathbf{y}_{\text{mis}} \end{aligned}$$

where $f(\mathbf{y}_{\text{mis}} \mid \mathbf{y}_{\text{obs}}, \boldsymbol{\theta}) = f(\mathbf{y}_{\text{mis}}, \mathbf{y}_{\text{obs}} \mid \boldsymbol{\theta})/F(\boldsymbol{\theta})$, the conditional distribution of $\mathbf{y}_{\text{mis}}$.

- **M-step.** Find $\boldsymbol{\theta}^{(t+1)}$ to maximize $Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^{(t)})$.

The E-step is derived from an "imputation" heuristic. Because we assume that the log-likelihood function is easy to compute once the missing data $\mathbf{y}_{\text{mis}}$ is given, it is appealing to simply "fill-in" a set of missing data and conduct a complete-data analysis. However, the simple fill-in idea is incorrect because it underestimates the variability caused by the missing information. The correct approach is to average the log-likelihood over all the missing data. In general, the E-step considers all possible ways of filling in the missing data, computes the corresponding complete-data log-likelihood function, and then obtains $Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^{(t)})$ by averaging these functions according to the current "predictive density" of the missing data. The M-step then finds the maximum of the Q function.

It is instructive to consider the EM algorithm for the latent-class model of section 2.3.2. The observed values are $\mathbf{y}_{\text{obs}} = (y_1, \ldots, y_n)$, where $y_i = (y_{i1}, y_{i2})$ and y_{ij} is the ith person's answer to jth question. The missing data are $\mathbf{y}_{\text{mis}} = (z_1, \ldots, z_n)$, where z_i is the latent-class label of person i. Let $\boldsymbol{\theta} = (\theta_{0,1}, \theta_{1,1}, \theta_{0,2}, \theta_{1,2}, \gamma)$, where γ is the frequency of $z_i = 1$ in the population and $\theta_{k,l}$ is the probability of a type-k person saying "yes" to the lth question. Then the complete-data likelihood is

$$\begin{aligned} f(\mathbf{y}_{\text{mis}}, \boldsymbol{\theta}) &= p(\mathbf{y}_{\text{obs}} \mid \mathbf{y}_{\text{mis}}, \boldsymbol{\theta}) p(\mathbf{y}_{\text{mis}} \mid \boldsymbol{\theta}) \\ &= \prod_{i=1}^{n} \left[\prod_{k=1}^{2} \{\theta_{z_i,k}^{y_{ik}} (1 - \theta_{z_i,k})^{1-y_{ik}}\} \gamma^{z_i} (1-\gamma)^{1-z_i} \right] \end{aligned}$$

The E-step requires us to average over all label imputations. Thus, $Q(\boldsymbol{\theta}\mid\boldsymbol{\theta}^{(t)})$ is equal to

$$E_t\left[\sum_{i=1}^{n}\left\{\sum_{k=1}^{2}\{y_{ik}\log\theta_{z_i,k}+(1-y_{ik})\log(1-\theta_{z_i,k})\}\right.\right.$$
$$\left.\left.+z_i\log\gamma+(1-z_i)\log(1-\gamma)\right\}\,\middle|\,\mathbf{y}_{\text{obs}}\right]$$

where the expectation sign means that we need to average out each z_i according to its "current" predictive probability distribution

$$\tau_i\equiv p(z_i=1\mid\mathbf{y}_{\text{obs}},\boldsymbol{\theta}^{(t)})=\frac{\gamma^{(t)}\theta_{1y_i}^{(t)}}{\gamma^{(t)}\theta_{1y_i}^{(t)}+(1-\gamma^{(t)})\theta_{0y_i}^{(t)}}$$

Hence, in the E-step, we simply fill-in a probabilistic label for each person, which gives

$$Q(\boldsymbol{\theta}\mid\boldsymbol{\theta}^{(t)})=\sum_{m=0}^{1}\sum_{k=1}^{2}\left(\sum_{i:\,y_{ik}=1}\tau_i^m(1-\tau_i)^{1-m}\log\theta_{m,k}\right.$$
$$\left.+\sum_{i:\,y_{ik}=0}\tau_i^m(1-\tau_i)^{1-m}\log(1-\theta_{m,k})\right)$$
$$+\left(\sum_{i=1}^{n}\tau_i\right)\log\gamma+\left(\sum_{i=1}^{n}(1-\tau_i)\right)\log(1-\gamma)$$

Although the above expression looks overwhelming, it is in fact quite simple and the M-step simply updates the parameters as $\gamma^{(t+1)}=\sum_{i=1}^{n}\tau_i/n$ and

$$\theta_{m,k}^{(t+1)}=\frac{\sum_{i:\,y_{ik}=1}\tau_i^m(1-\tau_i)^{1-m}}{\sum_{i:\,y_{ik}=1}\tau_i^m(1-\tau_i)^{1-m}+\sum_{i:\,y_{ik}=0}\tau_i^m(1-\tau_i)^{1-m}}.$$

There are three main advantages of the EM algorithm: (1) it is numerically stable (no inversion of a Hessian matrix); (2) each iteration of the algorithm strictly increases the value of the objective function unless it has reached a local optima; and (3) each step of the algorithm has an appealing statistical interpretation. For example, the E-step can often be seen as "imputing" the missing data and the M-step can be viewed as the estimation of the parameter value in lights of the current imputation.

The idea of iterating between filling-in missing data and updating estimate of the parameter has been around much longer than the EM algorithm. But Dempster et al. (1977) provided the first general and mathematically correct formulation of this intuitive idea (see Meng and van Dyk 1997 and discussions therein for an overview of recent advances of the EM algorithm).

2.4.2 Monte Carlo and Bayesian Analysis

As we have mentioned previously, the Bayesian analysis of a statistical problem can be made based on the joint posterior distribution of *all* unknown variables:

$$p(\boldsymbol{\theta}, \mathbf{y}_{\text{mis}} \mid \mathbf{y}_{\text{obs}}) = \frac{p(\mathbf{y}_{\text{obs}}, \mathbf{y}_{\text{mis}} \mid \boldsymbol{\theta}) f_0(\boldsymbol{\theta})}{\iint p(\mathbf{y}_{\text{obs}}, \mathbf{y}'_{\text{mis}} \mid \boldsymbol{\theta}') f_0(\boldsymbol{\theta}')\, d\mathbf{y}'_{\text{mis}}\, d\boldsymbol{\theta}'} \tag{2.6}$$

Note that this joint distribution is almost completely known—except for the denominator, which is often called the *normalizing constant* (or the *partition function* in physics). Suppose, for example, we are interested only in estimating the first component θ_1 of $\boldsymbol{\theta}$, say. We may need to evaluate its posterior mean (and perhaps other characteristics):

$$\begin{aligned} E(\theta_1 \mid \mathbf{y}) &= \iint \theta_1 p(\boldsymbol{\theta}, \mathbf{y}_{\text{mis}} \mid \mathbf{y}_{\text{obs}})\, d\mathbf{y}_{\text{mis}}\, d\boldsymbol{\theta} \\ &= \frac{\iint \theta_1 p(\mathbf{y}_{\text{obs}}, \mathbf{y}_{\text{mis}} \mid \boldsymbol{\theta}) f_0(\boldsymbol{\theta})\, d\mathbf{y}_{\text{mis}}\, d\boldsymbol{\theta}}{\iint p(\mathbf{y}_{\text{obs}}, \mathbf{y}'_{\text{mis}} \mid \boldsymbol{\theta}') f_0(\boldsymbol{\theta}')\, d\mathbf{y}'_{\text{mis}}\, d\boldsymbol{\theta}'} \end{aligned} \tag{2.7}$$

Neither the numerator nor the denominator in equation (2.7) is easy to compute in practice.

If, however, we can generate a random sample $(\mathbf{y}_{\text{mis}}^{(1)}, \boldsymbol{\theta}^{(1)}), \ldots, (\mathbf{y}_{\text{mis}}^{(m)}, \boldsymbol{\theta}^{(m)})$, either independently or dependently (as in a Markov chain), from the joint posterior distribution (2.6), then we can approximate the marginal posterior distribution of θ_1 by the histogram of the first component, $\theta_1^{(j)}$, of each $\boldsymbol{\theta}^{(j)}$, and approximate (2.7) by the Monte Carlo sample average

$$\tilde{\theta}_1 = \frac{1}{m}(\theta_1^{(1)} + \cdots + \theta_1^{(m)}) \tag{2.8}$$

2.4.3 Simple Monte Carlo Techniques

To begin with basic ideas, we describe two simple algorithms for generating random variables from a given distribution. As a starting point, we assume that independent *uniform* (in region [0,1]) random variables can be produced satisfactorily. Algorithms

that serve this purpose are called *random number generators*. In fact, this task is not as simple it looks, and the interested reader is encouraged to study further on this topic (Marsaglia and Zaman 1993).

Inversion Method When we have available the cumulative distribution function (cdf) for a one-dimensional target distribution $\pi(\mathbf{x})$, we can implement the following procedure.

- Draw $U \sim \text{Unif}[0,1]$;
- Compute $\mathbf{x} = F^{-1}(U)$, where $F^{-1}(u) = \inf\{x; F(x) \geq u\}$.

Then $\mathbf{x}$ so produced must follow π. The interested reader can try to prove this fact. However, because many distributions (e.g., Gaussian) do not have a closed-form cdf, it is often difficult to directly apply the inversion method. To overcome this difficulty, von Neumann (1951) invented the ingenious *rejection method*, which can be applied very generally.

Rejection Method Suppose $l(\mathbf{x}) = c\pi(\mathbf{x})$ is known (but c may be unknown) and we can find a sampling distribution $g(\mathbf{x})$ together with a constant M such that the envelope property, that is, $Mg(\mathbf{x}) \geq l(\mathbf{x})$ for all $\mathbf{x}$, is satisfied. Then we can apply the following procedure.

(1) Draw $\mathbf{x} \sim g(\mathbf{x})$ and compute the ratio $r = l(\mathbf{x})/Mg(\mathbf{x})$ (which should always be ≤ 1);

(2) Draw $U \sim \text{Unif}[0,1]$; accept and return $\mathbf{x}$ if $U \leq r$; reject $\mathbf{x}$ and go back to (a) if $U > r$.

To show that the accepted sample follows distribution π, we let I be the indicator function so that $I = 1$ if sample $\mathbf{X}$ drawn from $g(\ )$ is accepted, and $I = 0$, otherwise. Thus,

$$
\begin{aligned}
p(I = 1) &= \int p(I = 1 \mid \mathbf{X} = \mathbf{x}) g(\mathbf{x})\, d\mathbf{x} \\
&= \int \frac{c\pi(\mathbf{x})}{Mg(\mathbf{x})} g(\mathbf{x})\, d\mathbf{x} \\
&= \frac{c}{M}
\end{aligned}
$$

and

$$p(\mathbf{x} \mid I = 1) = \frac{c\pi(\mathbf{x})}{Mg(\mathbf{x})} g(\mathbf{x})/p(I = 1)$$
$$= \pi(\mathbf{x})$$

It is seen that the "success rate" for obtaining an accepted sample is c/M. Thus, the key to a successful application of the algorithm is to find a good trial distribution $g(\mathbf{x})$, which gives rise to a small M. Because it is usually difficult to find a good g-function in high-dimensional problems, the rejection method alone tends to be not very useful in difficult problems.

2.4.4 Markov Chain Monte Carlo Methods

Markov chain Monte Carlo (MCMC) is a class of algorithms for simulating random variables from a target distribution, $\pi(\mathbf{x})$, given up to a normalizing constant. A major advantage of these algorithms is their ability to "divide and conquer" a high-dimensional and complex problem. These algorithms serve our purpose well because in Bayesian analysis we want to draw random samples from the joint posterior distribution (2.6) without having to know its denominator. The basic idea behind MCMC algorithms is to design and simulate a Markov chain whose equilibrium distribution is exactly $\pi(\mathbf{x})$. Here we describe two methods for constructing such chains—the Metropolis algorithm and the Gibbs sampler—both being widely used in diverse fields. More versatile algorithms and their analyses can be found in Liu (2001).

Metropolis-Hastings Algorithm Let $\pi(\mathbf{x}) = c\exp\{-h(\mathbf{x})\}$ be the target distribution with unknown constant c. Metropolis et al. (1953) introduced the fundamental idea of Markov chain sampling and prescribed the first general construction of such a chain. Hastings (1970) later provided an important generalization. Starting with any configuration $\mathbf{x}^{(0)}$, the M-H algorithm evolves from the current state $\mathbf{x}^{(t)} = \mathbf{x}$ to the next state $\mathbf{x}^{(t+1)}$ as follows:

- Propose a new state $\mathbf{x}'$ that can be viewed as a small and random "perturbation" of the current state. More precisely, $\mathbf{x}'$ is generated from a *proposal* function $T(\mathbf{x}^{(t)} \to \mathbf{x}')$ (i.e., it is required that $T \geq 0$ and $\sum_{\text{all } \mathbf{y}} T[\mathbf{x} \to \mathbf{y}] = 1$ for all $\mathbf{x}$) determined by the user.
- Compute the Metropolis ratio

$$r(\mathbf{x}, \mathbf{x}') = \frac{\pi(\mathbf{x}')T(\mathbf{x}' \to \mathbf{x})}{\pi(\mathbf{x})T(\mathbf{x} \to \mathbf{x}')} \tag{2.9}$$

- Generate a random number $u \sim \text{Unif}[0,1]$;

—let $\mathbf{x}^{(t+1)} = \mathbf{x}'$ if $u \leq r(\mathbf{x}, \mathbf{x}')$;

—let $\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)}$ otherwise.

A more well-known form of the Metropolis algorithm is described as iterating the following steps: (1) a small random perturbation of the current configuration is made; (2) the "gain" (or loss) in an objective function (i.e., $-h[\mathbf{x}]$) resulting from this perturbation is computed; (2) a random number U is generated independently; and (4) the new configuration is accepted if $\log(U)$ is smaller than or equal to the "gain," and is rejected otherwise. The well-known *simulated annealing* algorithm (Kirkpatrick et al. 1983) is built upon this basic Metropolis iteration by adding an adjustable exponential scaling parameter to the objective function (i.e., $\pi[\mathbf{x}]$ is scaled to $\pi^{\alpha}[\mathbf{x}]$ and *alpha* $\rightarrow 0$).

Metropolis et al. (1953) restricted their choices of the "perturbation" function to be the symmetric ones. That is, the chance of proposing $\mathbf{x}'$ from perturbing $\mathbf{x}$ is always equal to that of proposing $\mathbf{x}$ from perturbing $\mathbf{x}'$. Intuitively, this means that there is no "trend bias" at the proposal stage. Mathematically, this symmetry can be expressed as $T(\mathbf{x} \rightarrow \mathbf{x}') = T(\mathbf{x}' \rightarrow \mathbf{x})$. Hastings (1970) generalized the choice of T to all those that satisfies the property: $T(\mathbf{x} \rightarrow \mathbf{x}') > 0$ if and only if $T(\mathbf{x}' \rightarrow \mathbf{x}) > 0$. It is easy to see that the "actual" transition probability function resulting from the M-H transition rule is, for $\mathbf{x} \neq \mathbf{y}$,

$$A(\mathbf{x} \rightarrow \mathbf{y}) = T(\mathbf{x} \rightarrow \mathbf{y}) \min\{1, r(\mathbf{x}, \mathbf{y})\}$$

where $r(\mathbf{x}, \mathbf{y})$ is the Metropolis ratio as in (2.9). It is easy to see that

$$\pi(\mathbf{x})A(\mathbf{x} \rightarrow \mathbf{y}) = \min\{\pi(\mathbf{x})T(\mathbf{x} \rightarrow \mathbf{y}), \pi(\mathbf{y})T(\mathbf{y} \rightarrow \mathbf{x})\}$$

which is a symmetric function in $\mathbf{x}$ and $\mathbf{y}$. Thus, the *detailed balance* condition

$$\pi(\mathbf{x})A(\mathbf{x} \rightarrow \mathbf{y}) = \pi(\mathbf{y})A(\mathbf{y} \rightarrow \mathbf{x})$$

is satisfied by A. This condition then implies that π is the *invariant* distribution for the Metropolis-Hastings transition. That is,

$$\int \pi(\mathbf{x})A(\mathbf{x} \rightarrow \mathbf{y})\, d\mathbf{x} = \pi(\mathbf{y})$$

Heuristically, π can be seen as a "fixed point" under the M-H operation in the space of all distributions. It follows from the standard Markov chain theory that if the chain is *irreducible* (i.e., it is possible to go from anywhere to anywhere else in a finite number of steps), *aperiodic* (i.e., there is no parity problem), and not drifting away,

then in the long run the chain will settle at its invariant distribution (Neal 1993). The random samples so obtained eventually are like those drawn directly from π.

The Metropolis algorithm has been extensively used in statistical physics over the past 40 years and is the cornerstone of all MCMC techniques recently adopted and generalized in the statistics community. Another type of MCMC algorithm, the Gibbs sampler (Geman and Geman 1984), differs from the Metropolis algorithm in its extensive use of conditional distributions based on $\pi(\mathbf{x})$ for constructing Markov chain moves.

Gibbs Sampler Suppose $\mathbf{x} = (x_1, \ldots, x_d)$. In the Gibbs sampler, one randomly or systematically chooses a coordinate, say x_1, and then update its value with a new sample x_1' drawn from the conditional distribution $\pi(\cdot \mid \mathbf{x}_{[-1]})$, where $\mathbf{x}_{[-A]}$ refers to $\{x_j, j \in A^c\}$. Algorithmically, the Gibbs sampler can be implemented as follows:

Random Scan Gibbs Sampler. Suppose currently $\mathbf{x}^{(t)} = (x_1^{(t)}, \ldots x_d^{(t)})$. Then

- Randomly select i from $\{1, \ldots, d\}$ according to a given probability vector $(\alpha_1, \ldots, \alpha_d)$.
- Let $x_i^{(t+1)}$ be drawn from the conditional distribution $\pi(\cdot \mid \mathbf{x}_{[-i]}^{(t)})$, and let $x_{[-i]}^{(t+1)} = \mathbf{x}_{[-i]}^{(t)}$.

Systematic Scan Gibbs Sampler. Let the current state be $\mathbf{x}^{(t)} = (x_1^{(t)}, \ldots x_d^{(t)})$.

- For $i = 1, \ldots, d$, we draw $x_i^{(t+1)}$ from the conditional distribution

$$\pi(x_i \mid x_1^{(t+1)}, \ldots, x_{i-1}^{(t+1)}, x_{i+1}^{(t)}, \ldots, x_d^{(t)})$$

It is easy to check that *every* individual conditional update leaves π invariant. Suppose currently $\mathbf{x}^{(t)} \sim \pi$. Then $\mathbf{x}_{[-i]}^{(t)}$ follows its marginal distribution under π. Thus,

$$\pi(x_i^{(t+1)} \mid \mathbf{x}_{[-i]}^{(t)}) \times \pi(\mathbf{x}_{[-i]}^{(t)}) = \pi(x_i^{(t+1)}, \mathbf{x}_{[-i]}^{(t)})$$

implying that the joint distribution of $(\mathbf{x}_{[-i]}^{(t)}, x_i^{(t+1)})$ is unchanged at π after one update.

The Gibbs sampler's popularity in statistics community stems from its extensive use of *conditional distributions* in each iteration. Tanner and Wong's (1987) data augmentation first linked the Gibbs sampling structure with missing data problems and the EM algorithm. Gelfand and Smith (1990) further popularized the method by pointing out that the conditionals needed in Gibbs iterations are commonly available in many Bayesian and likelihood computations.

Under regularity conditions, one can show that the Gibbs sampler chain converges geometrically and its convergence rate is related to how the variables correlate with each other. Therefore, grouping highly correlated variables together in the Gibbs update can greatly speed up the sampler (Liu 1994).

Other Techniques A main problem with all MCMC algorithms is that they may, for some problems, move very slowly in the configuration space or may be trapped in the region of a local mode. This phenomena is generally called *slow-mixing* of the chain. When chain is slow-mixing, estimation based on the resulting Monte Carlo samples becomes very inaccurate. Some recent techniques suitable for designing more efficient MCMC samplers in bioinformatics applications include simulated tempering (Marinari and Parisi 1992), parallel tempering (Geyer 1991), multicanonical sampling (Berg and Neuhaus 1992), multiple-try method (Liu et al. 2000), and evolutionary Monte Carlo (Liang and Wong 2000). These and some other techniques are summarized in Liu 2001.

2.5 Compositional Analysis of a DNA Sequence

Suppose our observation is a DNA sequence, $R = (r_1, r_2, \ldots, r_n)$, and we are interested in understanding various aspects of it, such as its general compositions, dependence between neighboring base pairs, regions with different statistical characteristics (e.g., G-C rich regions), repeated short sequence patterns, and so on. In this and the next sections we show how progressively complex statistical models can be developed to address these scientific questions. Note that the problem setting is very general because a dataset of multiple sequences can always be regarded as a single "super sequence" by joining all the individual sequences.

2.5.1 Multinomial Modeling

The simplest statistical model for a DNA sequence is, as we discussed in section 2.3.1, the iid multinomial model, in which each r_i is assumed to be independently generated according to probability vector $\boldsymbol{\theta} = (\theta_a, \ldots, \theta_t)$. The likelihood function of $\boldsymbol{\theta}$ is then $L(\boldsymbol{\theta} \mid R) = \theta_a^{n_a} \ldots \theta_t^{n_t}$, where $\mathbf{n} = (n_a, \ldots, n_t)$ is the vector of counts of the four types of nucleotides. Vector $\hat{\boldsymbol{\theta}} = (n_a/n, \ldots, n_t/n)$ maximizes $L(\boldsymbol{\theta} \mid R)$ and is the MLE of $\boldsymbol{\theta}$. The distribution of $n\hat{\boldsymbol{\theta}}$ under hypothetical replications is Multinom$(n; \boldsymbol{\theta})$; hence, for example, $n\hat{\theta}_a \sim \text{Binom}(n, \theta_a)$. Inverting this relationship gives us an approximate confidence interval for θ_a.

With a Dirichlet($\boldsymbol{\alpha}$) prior (3), the posterior of $\boldsymbol{\theta}$ is Dirichlet($\mathbf{n} + \boldsymbol{\alpha}$) and

$$E(\theta \mid R) = \left(\frac{n_a + \alpha_a}{n + \alpha}, \ldots, \frac{n_t + \alpha_t}{n + \alpha}\right)$$

where $\alpha = \alpha_a + \cdots + \alpha_t$. This result is not that much different from the MLE. If one is interested in the posterior distribution of θ_a, say, an easy calculation gives us

$$\theta_a \mid R \sim \text{Beta}(n_a + \alpha_a, n + \alpha - n_a - \alpha_a)$$

2.5.2 Homogeneous Markov Model

A natural next-step model is the *Markov* model, in which one assumes that the observed sequence follows a Markov chain with transition matrix $P(r_i \rightarrow r_{i+1}) = \theta_{r_i, r_{i+1}}$, where the parameter vector is a 4×4 matrix

$$\boldsymbol{\theta} = \begin{pmatrix} \theta_{aa} & \cdots & \theta_{at} \\ \vdots & \ddots & \vdots \\ \theta_{ta} & \cdots & \theta_{tt} \end{pmatrix}$$

where each row sums to one. The MLE of each component, θ_{at}, say, in the parameter matrix is $n_{at}/n_{a\cdot}$, where n_{at} is the total count of neighboring AT pairs in the sequence and $n_{a\cdot} = n_{aa} + \cdots + n_{at}$ is the total count of A's excluding the first bp r_1. When a conjugate prior is used, a similar procedure to that for the multinomial model gives us the posterior distribution of $\boldsymbol{\theta}$, which is a product of four (one for each row of $\boldsymbol{\theta}$) independent Dirichlet distributions.

2.5.3 A Hidden Markov Model

Let us now consider a model that can accommodate compositional heterogeneity in DNA sequences. For this we can think of sequence R as consisting of different segments, and the sequence composition are homogeneous within each segment. Based on this heuristics, Liu and Lawrence (1999) proposed and analyzed a Bayesian segmentation model. Another model, as first proposed by Churchill (1989), is based on the HMM structure shown in figure 2.1.

In this HMM model, we assume that the hidden layer $\mathbf{h} = (h_0, h_1, \ldots, h_n)$ is a Markov chain Each h_i, for example, may have two possible states where $h_i = 0$ implies that the corresponding r_i follows one compositional model, Multinom$(\boldsymbol{\theta}_0)$, and $h_i = 1$ indicates that $r_i \sim$ Multinom$(\boldsymbol{\theta}_1)$. Here $\boldsymbol{\theta}_k = (\theta_{ka}, \ldots, \theta_{kt})$. A 2×2 transition matrix, $\boldsymbol{\tau} = (\tau_{kl})$, where $\tau_{kl} = P(h_i = k \rightarrow h_{i+1} = l)$, dictates the generation of $\mathbf{h}$. Krogh et al. (1994) have developed a similar model to predict protein coding regions in *E. coli* genome.

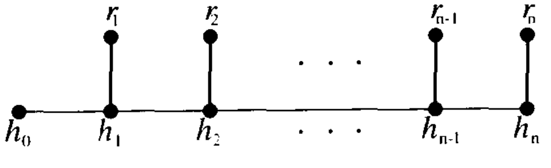


Figure 2.1
A graphical illustration of the hidden Markov model.

Let $\boldsymbol{\theta} = (\boldsymbol{\theta}_0, \boldsymbol{\theta}_1, \boldsymbol{\tau})$. The likelihood function of $\boldsymbol{\theta}$ under this HMM is

$$L(\boldsymbol{\theta}_0, \boldsymbol{\theta}_1, \boldsymbol{\tau} \mid R) = \sum_{\mathbf{h}} p(R \mid \mathbf{h}, \boldsymbol{\theta}_0, \boldsymbol{\theta}_1) p(\mathbf{h} \mid \boldsymbol{\tau})$$

$$= \sum_{\mathbf{h}} p_0(h_0) \prod_{i=1}^{n} p(r_i \mid h_i, \boldsymbol{\theta}) p(h_i \mid h_{i-1}, \boldsymbol{\tau})$$

$$= \sum_{\mathbf{h}} p_0(h_0) \prod_{i=1}^{n} \theta_{h_i r_i} \tau_{h_{i-1} h_i}$$

where h_0 is assumed to follow a known distribution p_0. For a given set of parameter values, we can find the exact value of this likelihood function via a recursive summation method as described in equation (2.10), below.

However, finding the MLE of $\boldsymbol{\theta}$ is still nontrivial. One possible approach is to maximize L by a Newton-Raphson's method in which the first and the second derivatives of L can all be computed recursively. But this method may be unstable because $\boldsymbol{\theta}$'s dimensionality is a bit too high (the Hessian is a 9×9 matrix). A more stable alternative is the EM algorithm, which involves iterations of the following two steps.

- *E-step.* Compute the Q-function:

$$Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^{(t)}) = E\left[\sum_{i=1}^{n} \log\{\theta_{h_i r_i} \tau_{h_{i-1} h_i}\} \mid R, \boldsymbol{\theta}^{(t)}\right]$$

$$= \sum_{i=1}^{n} \left[\sum_{h_i} \sum_{h_{i-1}} \{\log \theta_{h_i r_i} + \log \tau_{h_{i-1} h_i}\} P_i^{(t)}(h_{i-1}, h_i)\right]$$

$$= \sum_{k=0}^{1} \sum_{j=a}^{t} n_{kj}^{(t)} \log \theta_{kj} + \sum_{k=0}^{1} \sum_{l=0}^{1} m_{kl}^{(t)} \log \tau_{kl}$$

Here $P_i^{(t)}(h_{i-1}, h_i) = p(h_{i-1}, h_i \mid R, \boldsymbol{\theta}^{(t)})$, the marginal posterior distribution of (h_{i-1}, h_i) when the parameter takes value $\boldsymbol{\theta}^{(t)}$. This quantity can be obtained by using the B-function defined in equation (2.16) and a procedure similar to the computation of equation (2.17). The $P_i^{(t)}$ can be derived by a recursive procedure similar to equation (2.10). The $n_{kj}^{(t)}$ and the $m_{kl}^{(t)}$ are the sums of the corresponding $P_i^{(t)}(k, l)$.

- *M-step.* Maximize the Q-function. It is obvious that the maximizer of $Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^{(t)})$ is

$$\theta_{kj}^{(t+1)} = n_{kj}^{(t)}/n_{k\cdot}^{(t)} \quad \text{and} \quad \tau_{kl}^{(t+1)} = m_{kl}^{(t)}/m_{k\cdot}^{(t)}$$

in which $n_{k\cdot}^{(t)} = n_{ka}^{(t)} + \cdots + n_{kt}^{(t)}$ and $m_{k\cdot}^{(t)} = m_{k0}^{(t)} + m_{k1}^{(t)}$

To avoid being trapped at a singular point corresponding to zero count of certain base type, we may want to give a nonzero *pseudo*-count to each type.

A Bayesian analysis of this problem is also feasible. With a prior distribution $f_0(\boldsymbol{\theta})$, which may be a product of three independent Dirichlet distributions, we have the joint posterior of all unknowns:

$$p(\boldsymbol{\theta}, \mathbf{h} \mid R) \propto p(R \mid \mathbf{h}, \boldsymbol{\theta}) p(\mathbf{h} \mid \boldsymbol{\theta}) f_0(\boldsymbol{\theta})$$

In order to get the marginal posterior of $\boldsymbol{\theta}$, we may implement a special Gibbs sampler, data augmentation, which iterates the following steps:

- *Imputation:* draw $\mathbf{h}^{(t+1)} \sim p(\mathbf{h} \mid R, \boldsymbol{\theta}^{(t)})$;
- *Posterior Sampling:* draw $\boldsymbol{\theta}^{(t+1)} \sim p(\boldsymbol{\theta} \mid R, \mathbf{h}^{(t+1)})$.

The imputation step needs to draw a path, $\mathbf{h}$, from its *posterior* distribution with a given parameter value. Its implementation requires a recursive method for summing up all the contributions from h_0 to h_n and then sampling backward. Thus, this method is very similar to dynamic programming and is sometimes called the *forward-backward* method. More precisely, this distribution can be written as

$$\begin{aligned} p(\mathbf{h} \mid R, \boldsymbol{\theta}) &= cp(\mathbf{h}, R \mid \boldsymbol{\theta}) \\ &= cp(R \mid \mathbf{h}, \boldsymbol{\theta}) p(\mathbf{h} \mid \boldsymbol{\theta}) \\ &= cp_0(h_0) \prod_{i=1}^{n} \{ p(r_i \mid h_i) p(h_i \mid h_{i-1}) \} \\ &= cp_0(h_0) \prod_{i=1}^{n} (\theta_{h_i r_i} \tau_{h_{i-1} h_i}) \end{aligned}$$

where c is the normalizing constant, that is,

$$c^{-1} = \sum_{\mathbf{h}} \left\{ p_0(h_0) \prod_{i=1}^{n} (\theta_{h_i r_i} \tau_{h_{i-1} h_i}) \right\}$$

The key observation is that c, and also other required marginal distributions, can be computed exactly by a recursive method. Define $F_0(h) = p_0(h)$, and compute recursively

$$F_i(h) = \sum_{h_{i-1}=1}^{2} \{F_{i-1}(h_{i-1})\tau_{h_{i-1}h}\theta_{hr_i}\}, \quad \text{for } h = 0, 1 \tag{2.10}$$

At the end of the recursion we obtain $c^{-1} = F_n(0) + F_n(1)$ and

$$p(h_n \mid R, \boldsymbol{\theta}) = \frac{F_n(h_n)}{F_n(0) + F_n(1)} \tag{2.11}$$

In order to sample **h** properly, we draw h_n from distribution (2.11) and then draw h_i recursively backward from the distribution

$$p(h_i \mid h_{i-1}, R, \boldsymbol{\theta}) = \frac{F_i(h_i)\tau_{h_i h_{i+1}}}{F_i(0)\tau_{0h_{i+1}} + F_i(1)\tau_{1h_{i+1}}} \tag{2.12}$$

The posterior sampling step in the Gibbs sampler needs us to draw from the posterior distribution of $\boldsymbol{\theta}$ given **h** and R. This is a very simple task and only involves finding appropriate counts and sampling from the corresponding Dirichlet distributions. For example, $\boldsymbol{\theta}_0$ should be drawn from Dirichlet$(n_{0a} + \alpha_a, \ldots, n_{0t} + \alpha_t)$, where n_{0a}, say, is the counts of the r_i whose type is A and whose hidden state h_i is zero.

2.5.4 HMM with More than Two Hidden States

It is straightforward to extend the previous two-state HMM to a k-state HMM so as to analyze a sequence with regions of k different compositional types. In a k-state HMM, we will need a $k \times k$ transition matrix ($k(k-1)$ free parameters) to describe the transitions between the hidden Markov chain, and a probability vector $\boldsymbol{\theta}_j$ for each compositional type ($3k$ free parameters). The total number of free parameters is thus $k(k+2)$.

It is a nontrivial problem, however, to determine what value of k is proper for a given sequence R. A Bayesian model selection procedure as described in section 2.3.3 can be applied to resolve this issue. More precisely, we introduce a *model variable K*. For given $K = k$, we can fit a k-state HMM to the sequence and obtain the *model likelihood*

$$p(R \mid K = k) = \iint p_k(R \mid \mathbf{h}, \boldsymbol{\theta}) p_k(\mathbf{h} \mid \boldsymbol{\theta}) f_k(\boldsymbol{\theta})\, d\mathbf{h}\, d\boldsymbol{\theta}$$

where subscript k indicates that the employed distributions correspond to a k-state model. With a prior distribution $p_0(k)$ on K, we can derive the posterior distribution of K given the sequence. Although conceptually simple, this model selection procedure involves a difficult integral that is difficult to solve analytically. One often has

to resort to some special MCMC methods designed for estimating the ratio of normalizing constants (Liu 2001).

As an alternative to the HMM, Liu and Lawrence (1999) and Schmidler et al. (2000) describe a *segmentation* model based on the so-called *hidden semi-Markov model* (HSMM). Sequence segmentation models have been developed for many purposes in bioinformatics, including models for protein sequence hydrophobicity (Kyte and Dolittle 1982; Auger and Lawrence 1989), models for protein secondary structure (Schmidler et al. 2000), models for sequence complexity (Wootton 1994), and models for gene identification (Snyder and Stormo 1995; Burge and Karlin 1997). What is common to all these methods is that a single sequence is characterized by a series of models that only involve local properties. That is, we assume in this model that the sequence can be segmented into m parts, where m is unknown, and each segment is described by a "local" model. An advantage of this model is that a Bayesian method for determining the number of segments m is relatively easy (Liu and Lawrence 1999).

2.6 Find Repetitive Patterns in DNA Sequence

Similar to the objective of the previous section, our primary interest here is in the analysis of a single "super-sequence." Our focus, however, is one step further than the compositional analysis: we want to find repetitive motif elements in the sequence. The main motivation for this task is that repetitive patterns in biopolymer sequences often correspond to functionally or structurally important parts of these molecules. For example, repetitive patterns in noncoding regions of DNA sequences may correspond to a "regulatory motif" to which certain regulatory proteins bind so as to control gene expressions. The multiple occurrences of a regulatory motif in R is thus analogous to the multiple occurrences of a *word* in a long sentence. It is of interest to find out what this motif is and where it has occurred. What makes things worse, however, is that although the motif occurs in the sequence multiple times, no two occurrences are exactly identical. In other words, there are often some "typos" in each occurrence of the word. It is therefore rather natural for us to employ probabilistic models to handle this problem.

2.6.1 Block-Motif Model with IID Background

A simple model that conveys the basic idea of a motif that repeats itself with random variations is the block-motif model shown in figure 2.2. It was first developed in Liu et al. (1995) and has been employed to find subtle repetitive patterns, such as helix-

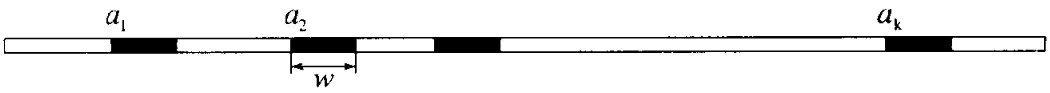


Figure 2.2
A graphical illustration of the repetitive motif model.

turn-helix structural motifs (Neuwald et al. 1995) or gene regulation motifs (Roth et al. 1998), in both protein and DNA sequences.

This model says that at unknown locations $A = (a_1, \ldots, a_K)$ there are repeated occurrence of a motif, so the sequence segments at these locations should look similar to each other. In another part of the sequence, called the *background*, the residues follow an independent multinomial model. Suppose the motif's width is w. We need $w+1$ probability vectors to describe the motif and the background: $\boldsymbol{\theta}_0 = (\theta_{0a}, \ldots, \theta_{0t})$ describe the base frequencies in the background; and each $\boldsymbol{\theta}_k$ describes the base frequency at position k of the motif. The matrix $\Theta = [\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_w]$ is called the *profile* matrix for the motif. We again use the generic notation $\boldsymbol{\theta}$ to denote the collection of all parameters, $(\boldsymbol{\theta}_0, \Theta)$.

With a Dirichlet prior Dirichlet($\boldsymbol{\alpha}$) for all the $\boldsymbol{\theta}_i$, we can obtain the Bayes estimates of the $\boldsymbol{\theta}_i$ very easily *if* we know the positions of the motif. To facilitate analysis, we introduce an indicator vector $\boldsymbol{I} = (I_1, \ldots, I_n)$ and treat it as *missing data*. An $I_i = 1$ means that position i is the start of a motif pattern, and $I_i = 0$ means otherwise. We assume a priori each I_i has a small probability p_0 to be equal to 1. With this setup, we can write down the joint posterior distribution:

$$p(\boldsymbol{\theta}, \boldsymbol{I} \mid R) \propto p(R \mid \boldsymbol{I}, \boldsymbol{\theta}) p(\boldsymbol{I} \mid \boldsymbol{\theta}) f_0(\boldsymbol{\theta}) \tag{2.13}$$

where

$$p(\boldsymbol{I} \mid \boldsymbol{\theta}) \propto \prod_{i=1}^{n} p_0^{I_i} (1 - p_0)^{1 - I_i}$$

If we do not allow overlapping motifs, we need to restrict that in $\boldsymbol{I}$ there are no pair $I_i = 1$ and $I_j = 1$ with $i - j < w$. Because the motif region is a very small fraction of the whole sequence, we may estimate $\boldsymbol{\theta}_0$ based on the whole sequence and treat it as known.

A simple Gibbs sampler algorithm can be designed to draw from this joint posterior distribution (Liu et al. 1995). More specifically, we can iterate the following steps:

• For a current realization of $\boldsymbol{\theta}$, we update each I_i, $i = 1, \ldots, n$, by a random draw from its conditional distribution, $p(I_i \mid \boldsymbol{I}_{[-1]}, R, \boldsymbol{\theta})$, where

$$\frac{p(I_i = 1 \mid \boldsymbol{I}_{[-i]}, R, \boldsymbol{\theta})}{p(I_i = 0 \mid \boldsymbol{I}_{[-i]}, R, \boldsymbol{\theta})} = \frac{p_0}{1 - p_0} \prod_{k=1}^{w} \left(\frac{\theta_{k, r_{i+k-1}}}{\hat{\theta}_{0 r_{i+k-1}}} \right) \tag{2.14}$$

Intuitively, this odds ratio is simply the "signal-to-noise" ratio.

• Based on the current value of $\boldsymbol{I}$, we update the profile matrix Θ column-by-column. That is, each θ_j, $j = 1, \ldots, w$, is drawn from an appropriate posterior Dirichlet distribution determined by $\boldsymbol{I}$ and R.

After a burn-in period (until the Gibbs sampler stabilizes), we continue to run the sampler for m iterations and use equation (2.8) to estimate the profile matrix Θ. The estimated Θ can then be used to scan the sequence to find the locations of the motif.

2.6.2 Block-Motif Model with a Markovian Background

Here the extra complication is that the motif can have a Markovian background. Thus, we need a 4×4 transition matrix, $B_0 = (\beta_{jj'})$, to describe the background. We also assume that the transition from the end of a motif to the next nonsite position follows the same Markov law. Because the total number of bp's that belong to a motif is a very small fraction of the total number of base pairs in R, we may estimate B_0 from the raw data directly, pretending that the whole sequence of R is homogeneous and governed by the transition matrix B_0. In this way, the transition probabilities can be estimated as $\hat{\beta}_{j_1 j_2} = n_{j_1 j_2}/n_{j_1}$, similar to that in section 2.5.2. We may then treat B_0 as a known parameter. The joint posterior distribution of $(\boldsymbol{\theta}, \boldsymbol{I})$ in this case differs from equation (2.13) only in the description of the residues in the background.

A Gibbs sampler very similar to the one described in section 2.6.1 can be implemented. The only difference is in the distribution $p(I_i \mid \boldsymbol{I}_{[-i]}, R, \boldsymbol{\theta})$ that is needed in the conditional update of $\boldsymbol{I}$. That is, conditional on $\boldsymbol{\theta}$, R, we slide through the whole sequence position-by-position to update I_i according to a random draw from $p(I_i \mid \boldsymbol{I}_{[-i]}, R, \boldsymbol{\theta})$, which satisfies

$$\frac{p(I_i = 1 \mid \boldsymbol{I}_{[-i]}, R, \boldsymbol{\theta})}{p(I_i = 0 \mid \boldsymbol{I}_{[-i]}, R, \boldsymbol{\theta})} = \frac{p_0}{1 - p_0} \prod_{k=1}^{w} \left(\frac{\theta_{k, r_{i+k-1}}}{\hat{\beta}_{r_{i+k-2} r_{i+k-1}}} \right)$$

For given $\boldsymbol{I}$, we update the profile matrix Θ in the same way as in section 2.6.1.

2.6.3 Block-Motif Model with Inhomogeneous Background

It has long been noticed that DNA sequences contain regions of distinctive compositions. As discussed in sections 2.5.3 and 2.5.4, a HMM can be employed to delineate a sequence with k types of regions. Suppose we decide to use a HMM to model sequence inhomogeneity. As we mentioned before, because the total motif residue is a very small fraction of the whole sequence, we may estimate the background model parameters directly by the methods in section 2.5.3, pretending that R does not contain any motifs. Then we treat these parameters as known at the estimated values. After these, there are two strategies to modify the odds ratio formula (2.14).

In the first strategy, we treat each position in the sequence as a "probabilistic bp" (i.e., having probabilities to be one of the four letters) and derive the frequency model from it. That is, we need to find $\theta_{ij}^* = p(r_i^* = j \mid R)$ for a future r_i^* and then treat residue r_i in the background as an independent observation from Multinom($\boldsymbol{\theta}_i^*$), with $\boldsymbol{\theta}_i^* = (\theta_{ia}^*, \ldots, \theta_{it}^*)$. But this computation is nontrivial because

$$\theta_{ij}^* = p(r_i^* = j \mid R) = \theta_{0j} p(h_i = 0 \mid R) + \theta_{1j} p(h_i = 1 \mid R) \tag{2.15}$$

where $p(h_i)$ can be computed via a recursive procedure similar to equation (2.10). More precisely, in addition to the series of forward functions F_i, we can define the backward functions B_i. Let $B_n(h) = \sum_{h_n} \tau_{hh_n} \theta_{h_n r_n}$, and let

$$B_k(h) = \sum_{h_k} \{\tau_{hh_k} \theta_{h_k r_k} B_{k+1}(h_k)\}, \quad \text{for } k = n-1, \ldots, 1 \tag{2.16}$$

Then we have

$$p(h_i = 1 \mid R) = \frac{F_i(1) B_{i+1}(1)}{F_i(1) B_{i+1}(1) + F_i(0) B_{i+1}(0)} \tag{2.17}$$

This is the *marginal* posterior distribution of h_i and can be used to predict whether position i is in state 1 or 0. Thus, in the Gibbs sampling algorithm we only need to modify the denominator of the right hand side of equation (2.14) to $\prod_{k=i}^{i+w-1} \theta_{kr_k}^*$.

In the second strategy, we seek to obtain the probability of the whole segment,

$$R_{[i:i+w-1]} \equiv (r_i, \ldots, r_{i+w-1})$$

conditional on the remaining part of the sequence, under the background HMM. Then we modify equation (2.14) accordingly. Clearly, compared with the first strategy, the second one is more faithful to the HMM assumption. The required probability evaluation can be achieved by a method similar to that in the first strategy.

More precisely,

$$\begin{aligned} p(R_{[i:i+w-1]} \mid R_{[1:i-1]}, R_{[i+w:n]}) &= \frac{p(R)}{p(R_{[1:i-1]}, R_{[i+w:n]})} \\ &= \frac{p(R)}{\sum_{\mathbf{h}} p(R_{[1:i-1]}, R_{[i+w:n]}, \mathbf{h})} \\ &= \frac{F_n(0) + F_n(1)}{\sum_{h_1,\ldots,h_w} F_i(h_1)\tau_{h_1h_2} \ldots \tau_{h_{w-1}h_w} B_{i-w}(h_w)} \end{aligned} \tag{2.18}$$

where the denominator can also be obtained via recursions.

2.6.4 Extension to Multiple Motifs

Previously, we have assumed that there is only one kind of motif in the sequence and the prior probability for each $I_i = 1$ is known as p_0. Both of these assumptions can be relaxed. Suppose we want to detect and align m different types of motifs of lengths $w_1, \ldots, w_m$, respectively, and each occurring unknown number of times in R. We can similarly introduce the indicator vector $\boldsymbol{I}$, where $I_i = j$ indicates that an element from motif j starts at position i, and $I_i = 0$ means that no elements start from position i. For simplicity, we only consider the independent background model.

Let $p(I_i = j) = \epsilon_j$, where $\epsilon_0 + \cdots + \epsilon_m = 1$, is an unknown probability vector. Given what is known about the biology of the sequences being analyzed, a crude guess k_j for the number of elements for motif j is usually possible. Let $k_0 = n - k_1 - \cdots - k_m$. We can represent this prior opinion about the number of occurrences of each type of elements by a Dirichlet distribution on $\boldsymbol{\varepsilon} = (\epsilon_0, \ldots, \epsilon_m)$, which has the form Dirichlet$(b_0, \ldots, b_m)$ with $b_j = J_0(k_j/n)$, where J_0 represents the "weight" (or "pseudo-counts") to be put on this prior belief. Then the same predictive updating approach as illustrated in section 2.6.1 can be applied. Precisely, the update formula (2.14) for $\boldsymbol{I}$ is changed to

$$\frac{\pi(I_i = j \mid \boldsymbol{I}_{[-i]}, R)}{\pi(I_i = 0 \mid \boldsymbol{I}_{[-i]}, R)} = \frac{\epsilon_j}{\epsilon_0} \prod_{k=1}^{w_j} \left(\frac{\theta^{(j)}_{kr_{i+k-1}}}{\theta_{0r_{i+k-1}}} \right)$$

where $\Theta^{(j)} = [\boldsymbol{\theta}^{(j)}_1, \ldots, \boldsymbol{\theta}^{(j)}_{w_j}]$ is the profile matrix for the jth motif. Conditional on $\boldsymbol{I}$, we can then update $\boldsymbol{\varepsilon}$ by a random sample from Dirichlet$(b_0 + n_0, \ldots, b_m + n_m)$, where n_j $(j > 0)$ is the number of motif type j found in the sequence, that is, the total number of i such that $I_i = j$, and $n_0 = n - \sum n_j$. More details can be found in Neuwald et al. 1995.

2.7 Discussion

As in classical statistics, optimization has been the primary tool in bioinformatics, in which point estimates of very high-dimensional objects obtained by dynamic programming or other clever computational methods are used. Characterizations of uncertainty in these estimates are mostly limited to simple significance test or completely ignored. The removal of nuisance parameters is also problematic, most frequently being the *profile likelihood* method in which the nuisance parameters are fixed at their best estimates. In comparison, the Bayesian method has no difficulties in these important aspects: the uncertainty in estimation is addressed by posterior calculations and the nuisance parameters are removed by summation and integration. When achievable, this class of principled approaches is particularly advantageous in treating bioinformatics problems (Liu et al. 1999; Zhu et al. 1998). In exchange for these advantages, however, one needs to set prior distributions and overcome computational hurdles, none of which are trivial in practice.

The most important limitation on the Bayesian method is the need for additional computational resources. Recursion-based Bayesian algorithms generally have time and space requirements of the same order as their dynamic programming counterparts, although the constants are generally much larger. With the availability of fast workstations with large memories, however, this moderate increase in computing need is not a serious difficulty for most applications. For those problems where there is no polynomial time solution, MCMC methods (and other Monte Carlo methods) provides alternative means to implement a full Bayesian analysis. Although the use of MCMC methods and recursive methods can ease some of the computational concerns, difficulties remain for the specification of sensible prior distributions.

Acknowledgments

This work was supported in part by the NSF grants DMS-9803649 and DMS-0074108. The author would like to thank Professor Donald B. Rubin and two anonymous referees for valuable suggestions.

References

Auger, I. E., and Lawrence, C. E. (1989). Algorithms for the optimal identification of segment neighborhoods. *Bull. Math. Biol.* 51(1): 39–54.

Baldi, P., Chauvin, Y., Hunkapiller, T., and McClure, M. A. (1994). Hidden Markov-models of biological primary sequence information. *Proc. Nat. Acad. Sci. of USA* 91(3): 1059–1063.

Berg, B. A., and Neuhaus, T. (1992). Multicanonical ensemble: A new approach to simulate first-order phase transition. *Physical Review Letters* 68: 9.

Bishop, M. J., and Thompson, E. A. (1986). Maximum-likelihood alignment of DNA-sequences. *J. Mol. Biol.* 190(2): 159–165.

Burge, C., and Karlin, S. (1997). Prediction of complete gene structures in human genomic DNA. *J. Mol. Biol.* 268(1): 78–94.

Cardon, L. R., and Stormo, G. D. (1992). Expectation maximization algorithm for identifying protein-binding sites with variable lengths from unaligned DNA fragments. *J. Mol. Biol.* 223(1): 159–170.

Churchill, G. A. (1989). Stochastic-models for heterogeneous DNA-sequences. *Bull. Math. Biol.* 51(1): 79–94.

Cox, D. R., and Hinkley, D. V. (1974). *Theoretical Statistics*. New York: Chapman & Hall.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via em algorithm. *Journal of the Royal Statistical Society Series B-Methodological* 39(1): 1–38.

Efron, B. (1979). Bootstrap methods: Another look at the jackknife. *Ann. Statist.* 7: 1–26.

Gelfand, A. E., and Smith, A. F. M. (1990). Sampling-based approaches to calculating marginal densities. *J. Am. Statist. Assoc.* 85: 398–409.

Gelman, A., Carlin, J. B., Stern, H. S., and Rubin, D. B. (1995). *Bayesian Data Analysis*. London: Chapman & Hall.

Geman, S., and Geman, D. (1984). Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. *IEEE Trans. Pattern Anal. Machine Intell.* 6: 721–741.

Geyer, C. (1991). Markov chain monte carlo maximum likelihood. In *Computing Science and Statistics: The 23rd Symposium on the Interface*, Keramigas, E., ed., 156–163. Fairfax: Interface Foundation.

Gilks, W. R., Richardson, S., and Spiegelhalter, D. J. (1998). *Markov Chain Monte Carlo in Practice*. Boca Raton, Fla.: Chapman & Hall.

Hastings, W. K. (1970). Monte Carlo sampling methods using Markov chains and their applications. *Biometrika* 57: 97–109.

Kass, R. E., and Raftery, A. E. (1995). Bayes factors. *J. Am. Statist. Assoc.* 90(430): 773–795.

Kirkpatrick, S., Gelatt, Jr. G., and Vecchi, M. (1983). Optimization by simulated annealing. *Science* 22(4598): 671–680.

Krogh, A., Brown, M., Mian, I. S., Sjolander, K., and Haussler, D. (1994). Hidden Markov models in computational biology: Applications to protein modeling. *J. Mol. Biol.* 235(5): 1501–1531.

Krogh, A., Mian, I. S., and Haussler, D. (1994). A hidden Markov model that finds genes in escherichia-coli DNA. *Nucl. Acids Res.* 22(22): 4768–4778.

Kyte, J., and Doolittle, R. F. (1982). A simple method for displaying the hydropathic character of a protein. *J. Mol. Biol.* 157(1): 105–132.

Lawrence, C. E., Altschul, S. F., Boguski, M. S., Liu, J. S., Neuwald, A. F., and Wootton, J. C. (1993). Detecting subtle sequence signals: A Gibbs sampling strategy for multiple alignment. *Science* 262(5131): 208–214.

Lawrence, C. E., and Reilly, A. A. (1990). An expectation maximization (em) algorithm for the idenification and characterization of common sites in unaligned biopolymer sequences. *Proteins* 7: 41–51.

Liang, F., and Wong, W. H. (2000). Evolutionary Monte Carlo: Applications to c_p model sampling and change point problem. *Statistica Sinica* 10(2): 317–342.

Liu, J. S. (1994). The collapsed Gibbs sampler in Bayesian computations with applications to a gene-regulation problem. *J. Am. Statist. Assoc.* 89(427): 958–966.

Liu, J. S. (2001). *Monte Carlo Strategies in Scientific Computing*. New York: Springer.

Liu, J. S., and Lawrence, C. E. (1999). Bayesian inference on biopolymer models. *Bioinformatics* 15(1): 38–52.

Liu, J. S., Liang, F., and Wong, W. H. (2000). The use of multiple-try method and local optimization in metropolis sampling. *J. Am. Statist. Assoc.* 95: 121–134.

Liu, J. S., Neuwald, A. F., and Lawrence, C. E. (1995). Bayesian models for multiple local sequence alignment and Gibbs sampling strategies. *J. Am. Statist. Assoc.* 90(432): 1156–1170.

Liu, J. S., Neuwald, A. F., and Lawrence, C. E. (1999). Markovian structures in biological sequence alignments. *J. Am. Statist. Assoc.* 94(445): 1–15.

Lowe, T. M., and Eddy, S. R. (1997). Trnascan-se: A program for improved detection of transfer RNA genes in genomic sequence. *Nucl. Acids Res.* 25(5): 955–964.

Marinari, E., and Parisi, G. (1992). Simulated tempering: A new Monte Carlo scheme. *Europhysiology Letters* 19: 451–458.

Marsaglia, G., and Zaman, A. (1993). Monkey tests for random number generators. *Comput. Math. Appl.* 26(9): 1–10.

Meng, X. L., and van Dyk, D. (1997). The em algorithm: An old folk-song sung to a fast new tune. *J. Royal Statist. Soc. B-Methodol.* 59(3): 511–540.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953). Equations of state calculations by fast computing machines. *J. Chem. Phys.* 21: 1087–1091.

Neal, R. M. (1993). Probabilistic inference using Markov chain monte carlo methods. *Tech. Rep., Comp. Sci. Dept., U. of Toronto* CRG-TR-93-1.

Neuwald, A. F., Liu, J. S., and Lawrence, C. E. (1995). Gibbs motif sampling: Detection of bacterial outer-membrane protein repeats. *Protein Sci.* 4(8): 1618–1632.

Neuwald, A. F., Liu, J. S., Lipman, D. J., and Lawrence, C. E. (1997). Extracting protein alignment models from the sequence database. *Nucl. Acids Res.* 25(9): 1665–1677.

Roth, F. P., Hughes, J. D., Estep, P. W., and Church, G. M. (1998). Finding DNA regulatory motifs within unaligned noncoding sequences clustered by whole-genome mRNA quantitation. *Nature Biotechnol* 16(10): 939–945.

Rubin, D. B. (1976). Inference and missing data. *Biometrika* 63(3): 581–590.

Schmidler, S. C., Liu, J. S., and Brutlag, D. L. (2000). Bayesian segmentation of protein secondary structure. *J. Comput. Biol.* 7(1–2): 233–248.

Snyder, E. E., and Stormo, G. D. (1995). Identification of protein-coding regions in genomic DNA. *J. Molec. Biol.* 248(1): 1–18.

Tanner, M., and Wong, W. H. (1987). The calculation of posterior distributions by data augmentation. *J. Am. Statist. Assoc.* 82: 528–550.

Thorne, J. L., Kishino, H., and Felsenstein, J. (1991). An evolutionary model for maximum-likelihood alignment of DNA sequences. *J. Molec. Evol.* 33(2): 114–124.

von Neumann, J. (1951). Various techniques used in connection with random digits. *Natl. Bureau Standards Appl. Math. Series* 12: 36–38.

Wootton, J. C. (1994). Nonglobular domains in protein sequences: Automated segmentation using complexity-measures. *Comput. Chem.* 18(3): 269–285.

Zhu, J., Liu, J. S., and Lawrence, C. E. (1998). Bayesian adaptive sequence alignment algorithms. *Bioinformatics* 14(1): 25–39.

Zuker, M. (1989). Computer-prediction of RNA structure. *Methods Enzymol.* 180: 262–288.

3 Bio-Sequence Comparison and Applications

Xiaoqiu Huang

3.1 Introduction

The structure of a genome is a linear sequence of nucleotides that encodes genes and regulatory elements. Genes are homologous if they are related by divergence from a common ancestor (Attwood 2000). Homologous genes perform the same or similar functions. The sequences of homologous genes in related organisms are usually similar. For example, the sequences of homologous genes in humans and mice are 85 percent similar on average (Makalowski et al. 1996). If a new genomic DNA sequence is very similar to the sequence of a gene whose function is known, it is very likely that the genomic DNA sequence contains a gene and its function is similar to the function of the known gene. If a new genomic DNA sequence is highly similar to a cDNA sequence, then the genomic DNA sequence contains a gene and the structure of the gene can be found by aligning the two sequences. Thus methods for comparing sequences are very useful for understanding the structures and functions of genes in a genome. This chapter focuses on methods for comparing two sequences, which often serve as a basis for multiple sequence comparison methods, a topic for the next chapter.

In the first part of this chapter, we describe algorithms for comparing two sequences. We present a global alignment algorithm for comparing two sequences that are entirely similar. We give a local alignment algorithm for comparing sequences that contain locally similar regions. We also describe efficient computational techniques for comparing long sequences. In the second part, we consider a general problem of comparing two sets of sequences. Every sequence in one set is compared with every sequence in the other set. We describe an efficient algorithm for this problem. In the third part, we present four applications to illustrate that sequence alignment programs are useful for the analysis of DNA and protein sequences. In the last part, we provide two directions for developments of new and improved sequence comparison methods.

3.2 Global Alignment

In this section, we first define a global alignment model. Then we describe a dynamic programming algorithm for computing an optimal global alignment of two sequences. Next we present a linear-space algorithm for computing an optimal global alignment. Finally we look at a way of reducing the time requirements of the algorithms.

3.2.1 An Alignment Model

A similarity relationship between two sequences, A and B, can be represented by an alignment of two sequences, an ordered list of pairs of letters of A and B. The alignment consists of substitutions, deletion gaps, and insertion gaps. A substitution pairs a letter of A with a letter of B. A substitution is a match if the two letters are identical, and a mismatch otherwise. A deletion gap is a gap where letters of A correspond to no letter of B, and an insertion gap is a gap where letters of B correspond to no letter of A. The length of a gap is the number of letters involved. Deletion and insertion gaps are defined with regard to transformation of sequence A into sequence B. An alignment of A and B shows a way to transform A into B, where a letter of A is replaced by a letter of B in every substitution, the letters of A in every deletion gap are deleted, and the letters of B in every insertion gap are inserted.

Below is an alignment of two DNA sequences, AGCTACGTACACTACC and AGCTATCGTACTAGC. This alignment contains 13 matches, one mismatch, an insertion gap of length 1, and a deletion gap of length 2.

AGCTA–CGTACACTACC

AGCTATCGTAC––TAGC

The similarity of an alignment is measured by a numerical number. Let $\sigma(a, b)$ be the score of a substitution involving letters a and b. Let numbers q and r be gap-open and gap-extension penalties, respectively. The numbers q and r are nonnegative. The score of a gap of length k is $-(q + k \times r)$. Values for the parameters σ, q and r are specified by the user. A letter-independent substitution table is usually used for comparison of DNA sequences. For example, each match is given a score of 10 and each mismatch a score of -20. Possible values for q and r are 40 and 2, respectively, for DNA sequences. A letter-dependent substitution table such as PAM250 (Dayhoff et al. 1978) and BLOSUM62 (Henikoff and Henikoff 1992) is usually used for comparison of protein sequences. Possible values for q and r are 10 and 2, respectively, for proteins. The similarity score of an alignment is just the sum of scores of each substitution and each gap in the alignment. The score of the example alignment given above is 24 using the given set of values for DNA sequences. An optimal (global) alignment of two sequences A and B is an alignment of A and B with the maximum score.

3.2.2 A Dynamic Programming Algorithm

Let $A = a_1a_2 \ldots a_m$ and $B = b_1b_2 \ldots b_n$ be two sequences of lengths m and n. A technique called dynamic programming in computer science is used to compute an

optimal global alignment of A and B. Let $A_i = a_1a_2 \ldots a_i$ and $B_j = b_1b_2 \ldots b_j$ be initial segments of lengths i and j of A and B. In this technique, a matrix S is introduced: $S(i, j)$ is the maximum score of all alignments of A_i and B_j. Thus $S(m, n)$ is the score of an optimal alignment of A and B. To compute the matrix S efficiently, two additional matrices, D and I, are introduced. Let $D(i, j)$ (D for deletion) be the maximum score of all alignments of A_i and B_j that end with a deletion gap. Let $I(i, j)$ (I for insertion) be the maximum score of all alignments of A_i and B_j that end with an insertion gap.

First consider how to compute $S(i, j)$. For $i > 0$ and $j > 0$, let $P(A_i, B_j)$ denote an alignment of A_i and B_j with the maximum score $S(i, j)$, that is, an optimal alignment of A_i and B_j. The last aligned pair of $P(A_i, B_j)$ has to be one of the following aligned pairs: a substitution pair (a_i, b_j), a deletion pair $(a_i, -)$, or an insertion pair $(-, b_j)$. If the last aligned pair of $P(A_i, B_j)$ is a substitution pair (a_i, b_j), then the portion of $P(A_i, B_j)$ before the last substitution pair (a_i, b_j) is an alignment of A_{i-1} and B_{j-1} with the maximum score $S(i-1, j-1)$, because $P(A_i, B_j)$ is an alignment of A_i and B_j with the maximum score. In this case, alignment $P(A_i, B_j)$ consists of alignment $P(A_{i-1}, B_{j-1})$ and the substitution pair (a_i, b_j). Thus the score of $P(A_i, B_j)$ is equal to the score of $P(A_{i-1}, B_{j-1})$ plus $\sigma(a_i, b_j)$, that is,

$$S(i, j) = S(i-1, j-1) + \sigma(a_i, b_j)$$

If the last aligned pair of $P(A_i, B_j)$ is a deletion pair $(a_i, -)$, then $P(A_i, B_j)$ is an alignment of A_i and B_j that ends with a deletion and has the maximum score. By the definition of $D(i, j)$, we have

$$S(i, j) = D(i, j)$$

Similarly, if the last aligned pair of $P(A_i, B_j)$ is an insertion pair $(-, b_j)$, we have

$$S(i, j) = I(i, j)$$

By the definitions of the matrices S, D, and I, the following inequalities are always true.

$$S(i, j) \geq S(i-1, j-1) + \sigma(a_i, b_j)$$

$$S(i, j) \geq D(i, j)$$

$$S(i, j) \geq I(i, j)$$

Thus we conclude that for $i > 0$ and $j > 0$,

$$S(i, j) = \max\{S(i-1, j-1) + \sigma(a_i, b_j), D(i, j), I(i, j)\}$$

Next consider how to compute $D(i, j)$. For $i > 0$ and $j > 0$, let $X(A_i, B_j)$ denote an alignment of A_i and B_j with the maximum score $D(i, j)$, which ends with a deletion pair $(a_i, -)$. Let $Y(A_{i-1}, B_j)$ denote the portion of $X(A_i, B_j)$ before the last pair. If $Y(A_{i-1}, B_j)$ ends with a deletion pair, then $Y(A_{i-1}, B_j)$ is an alignment of A_{i-1} and B_j with the maximum score $D(i-1, j)$, because $X(A_i, B_j)$ is a largest-scoring alignment of A_i and B_j that ends with a deletion gap. In other words, $X(A_i, B_j)$ consists of $X(A_{i-1}, B_j)$ and the deletion pair $(a_i, -)$. So we have

$$D(i, j) = D(i-1, j) - r$$

Note that the gap open penalty for the gap that includes the deletion pair $(a_i, -)$ is already included in $D(i-1, j)$. If $Y(A_{i-1}, B_j)$ does not end with a deletion pair, then $Y(A_{i-1}, B_j)$ is an alignment of A_{i-1} and B_j with the maximum score $S(i-1, j)$, because $X(A_i, B_j)$ is a largest-scoring alignment of A_i and B_j that ends with a deletion gap. In other words, $X(A_i, B_j)$ consists of $P(A_{i-1}, B_j)$ and the deletion pair $(a_i, -)$.

So we have

$$D(i, j) = S(i-1, j) - q - r$$

where the expression $-q - r$ is the score of the gap that consists only of the deletion pair $(a_i, -)$, and $S(i-1, j)$ is the score of an optimal alignment $P(A_{i-1}, B_j)$, which ends with a substitution pair or an insertion pair.

Appending the deletion pair $(a_i, -)$ to alignment $X(A_{i-1}, B_j)$ yields an alignment of A_i and B_j with score $D(i-1, j) - r$. Similarly, appending the deletion pair $(a_i, -)$ to alignment $P(A_{i-1}, B_j)$ yields an alignment of A_i and B_j with score $S(i-1, j) - q - r$. Because both alignments end with a deletion pair, we have by the definition of $D(i, j)$ that

$$D(i, j) \geq D(i-1, j) - r$$

$$D(i, j) \geq S(i-1, j) - q - r$$

Note that if $i = 1$, then $D(i-1, j)$ is undefined. We assume that $D(0, j)$ is given a value of $S(0, j) - q$, so that the inequality involving $D(i-1, j)$ still holds if $i = 1$. Combining all those inequalities together, we conclude that for $i > 0$ and $j > 0$,

$$D(i, j) = \max\{D(i-1, j) - r, S(i-1, j) - q - r\}$$

The recurrence for computing the matrix I for $i > 0$ and $j > 0$ is developed similarly.

The recurrences for the matrices S, D, and I for $i = 0$ or $j = 0$ can be easily developed. The recurrences for computing the matrices S, D, and I are summarized below.

$$S(0,0) = 0$$

$$S(i,0) = D(i,0) \quad \text{for } i > 0$$

$$S(0,j) = I(0,j) \quad \text{for } j > 0$$

$$S(i,j) = \max\{S(i-1,j-1) + \sigma(a_i,b_j), D(i,j), I(i,j)\} \quad \text{for } i > 0 \text{ and } j > 0$$

$$D(0,j) = S(0,j) - q \quad \text{for } j \geq 0$$

$$D(i,0) = D(i-1,0) - r \quad \text{for } i > 0$$

$$D(i,j) = \max\{D(i-1,j) - r, S(i-1,j) - q - r\} \quad \text{for } i > 0 \text{ and } j > 0$$

$$I(i,0) = S(i,0) - q \quad \text{for } i \geq 0$$

$$I(0,j) = I(0,j-1) - r \quad \text{for } j > 0$$

$$I(i,j) = \max\{I(i,j-1) - r, S(i,j-1) - q - r\} \quad \text{for } i > 0 \text{ and } j > 0$$

We present an alternative way for developing the recurrences for computing the three matrices. The alternative presentation is based on a grid graph of $m+1$ rows and $n+1$ columns in figure 3.1. Each entry in the graph consists of three nodes that correspond to the three matrices, respectively. For $i > 0$, each vertical edge from row $i-1$ to row i corresponds to a deletion pair $(a_i, -)$. For $j > 0$, each horizontal edge from column $j-1$ to column j corresponds to an insertion pair $(-, b_j)$. For $i > 0$ and $j > 0$, each diagonal edge from entry $(i-1, j-1)$ to entry (i,j) corresponds to a substitution pair (a_i, b_j). Each directed path from node S of entry $(0,0)$ to node S of entry (i,j) corresponds to an alignment of A_i and B_j. Assume that for any entry, the edge from node D to node S and the edge from node I to node S have a score of 0. The score of a path from node S of entry $(0,0)$ to a node of entry (i,j) is the sum of scores of every edge on the path. For any entry (i,j), define $S(i,j)$ to be the maximum score of paths from node S of entry $(0,0)$ to node S of entry (i,j), and define $D(i,j)$ and $I(i,j)$ similarly with respect to nodes D and I of entry (i,j). If there is no path from node S of entry $(0,0)$ to node D (or I) of entry (i,j), then $D(i,j)$ (or $I[i,j]$) can be set to $S(i,j) - q$ or any smaller value. This will simplify the presentation of a recurrence for computing the matrix D (or I) without causing any change to the value $D(i+1,j)$ (or $I[i,j+1]$).

Consider how to compute $S(i,j)$ for an internal entry (i,j) with $i > 0$ and $j > 0$. We partition the paths from from node S of entry $(0,0)$ to node S of entry (i,j) into three groups. One group contains all the paths that end with a diagonal edge of score $\sigma(a_i, b_j)$. The maximum score of paths in this group is $S(i-1,j-1) + \sigma(a_i,b_j)$.

Another group contains all the paths that end with a vertical edge. The maximum score of paths in this group is $D(i, j)$. The last group contains all the paths that end with a horizontal edge. The maximum score of paths in this group is $I(i, j)$. Thus $S(i, j)$ is the maximum of the three expressions, which is exactly identical to the recurrence for $S(i, j)$ with $i > 0$ and $j > 0$ given previously. Next consider how to compute $S(i, j)$ for a border entry (i, j) with $i = 0$ or $j = 0$. It is easy to see that $S(0, 0) = 0$. For $i = 0$ and $j > 0$, all the paths from node S of entry $(0, 0)$ to node S of entry (i, j) end with a horizontal edge, and hence $S(i, j)$ is equal to $I(i, j)$. Similarly, for $i > 0$ and $j = 0$, $S(i, j)$ is equal to $D(i, j)$. The recurrences for computing the matrices D and I can be developed in the same way.

The matrices can be computed in order of rows or columns. The value $S(m, n)$ is the score of an optimal alignment of A and B. If only the score $S(m, n)$ is needed, then two linear arrays and a few scalars are sufficient to carry out the computation. This algorithm is the result of a number of developments (Needleman and Wunsch 1970; Sellers 1974; Wagner and Fisher 1974; Waterman et al. 1976; Gotoh 1982).

An optimal alignment is found by a traceback procedure on the matrices S, D, and I. An optimal alignment corresponds to a path through the grid graph (figure 3.1) from node S of entry $(0, 0)$ to node S of entry (m, n). Let the current node be a newly determined node. An optimal path is recovered by repeatedly determining a node that is immediately before the current node on the path. Thus the pairs of an optimal alignment are generated in a reverse order, with the last pair produced first. Initially, the current node is node S of entry (m, n). First consider the case where the current node is node S of entry (i, j). The recurrences for S are used to determine a new node. If $i = 0$ and $j = 0$, then the traceback procedure terminates. Otherwise, if $j = 0$ or $S(i, j) = D(i, j)$, then the new node is node D of entry (i, j). Otherwise, if $i = 0$ or $S(i, j) = I(i, j)$, then the new node is node I of entry (i, j). Otherwise, the new node is node S of entry $(i - 1, j - 1)$ and a new pair for the optimal alignment is a substitution pair (a_i, b_j). Next consider the case where the current node is node D of entry (i, j). The recurrences for D are used to determine a new node. If $i = 1$ or $D(i, j) = S(i - 1, j) - q - r$, then the new node is node S of entry $(i - 1, j)$. Otherwise, then the new node is node D of entry $(i - 1, j)$. In each situation, a new pair for the optimal alignment is a deletion pair $(a_i, -)$. The case where the current node is node I of entry (i, j) is similarly handled.

The traceback procedure requires that the complete matrices be saved or additional information be saved to indicate how the value at each matrix entry is generated, which takes computer memory proportional to the product $m \times n$. Thus for two sequences of length 10,000, the algorithm takes computer memory in the order of 100,000,000 words. Because of the high computer memory requirement, only an

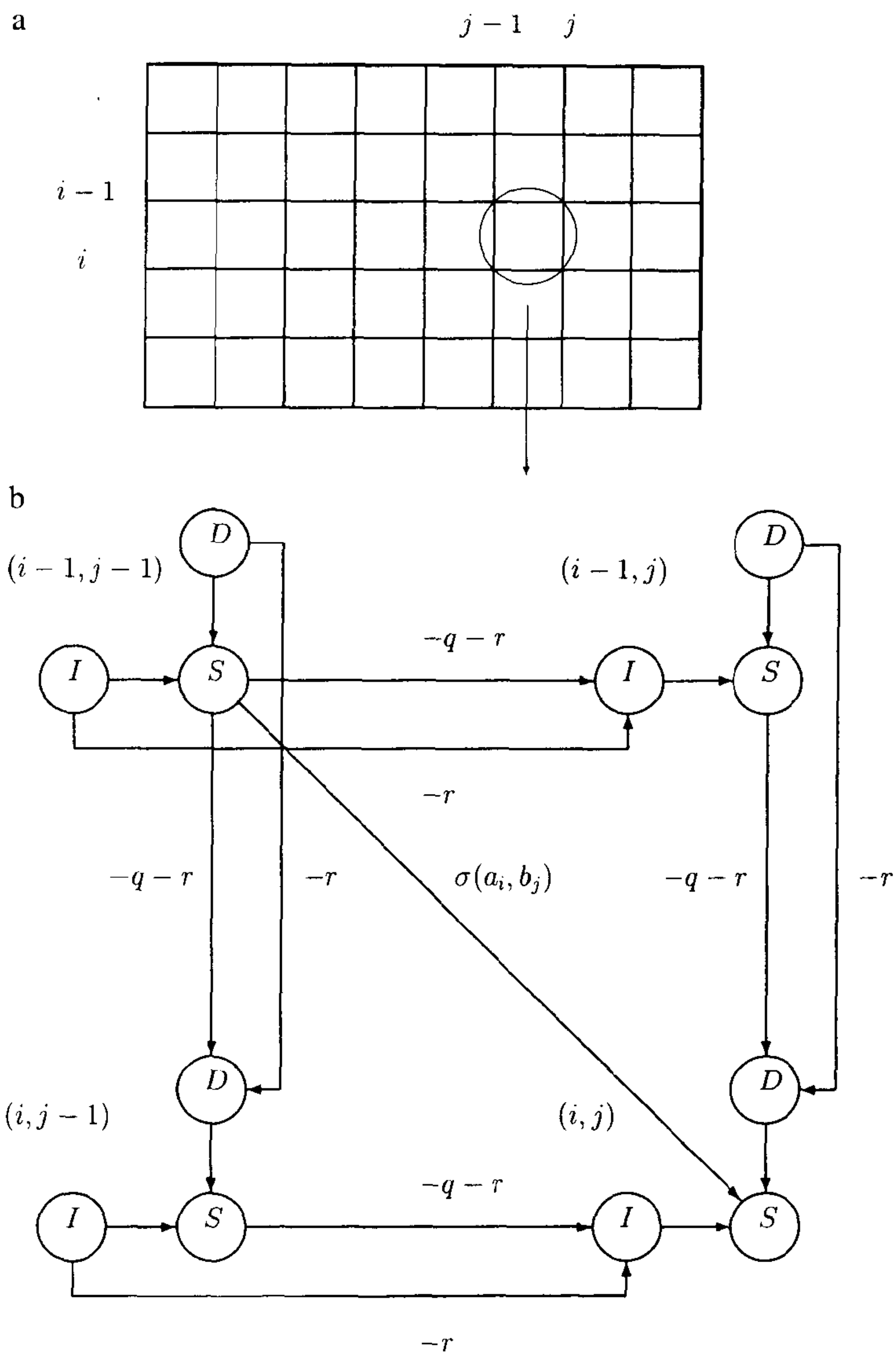


Figure 3.1
A grid graph. (a) An overview of the grid graph. (b) A detailed view of four adjacent entries in the graph. The edges from D to S and from I to S have a score of 0. The score of each remaining edge is shown next to the edge.

optimal alignment of two sequences of at most a few thousand letters can be constructed on an ordinary computer using this algorithm. The time requirement of the algorithm is also proportional to the product $m \times n$. For two sequences of length 10,000, it takes less than a minute to compute the matrix S on an ordinary workstation. Thus the space requirement of this algorithm is much more limiting than the time requirement.

3.2.3 A Linear-Space Algorithm

Hirschberg (1975) developed a linear-space algorithm for computing an optimal alignment of two sequences for the case $q = 0$. The algorithm takes computer memory in the order of $m + n$ and computer time in the order of $m \times n$. Because of the Hirschberg algorithm, computer memory is no longer a limiting factor for long sequences. In practice, the Hirschberg algorithm is even faster than the quadratic-space algorithm because an access to a linear array takes less time than an access to a quadratic array. As computers become faster, longer sequences can be aligned by the Hirschberg algorithm. Currently it takes about one hour on an ordinary workstation to produce an optimal alignment of two sequences of 100,000 letters. Myers and Miller (1988) generalized the algorithm of Hirschberg to handle the case where q is nonnegative.

The main idea of the space-efficient algorithm is to determine a middle pair of positions on an optimal alignment in linear space. Then the portions of the optimal alignment before and after the middle pair of positions are constructed recursively. Let *imid* be $\lfloor m/2 \rfloor$, where $\lfloor y \rfloor$ is the largest integer less than or equal to y. We develop an algorithm for finding a position *jmid* such that the pair of positions *imid* and *jmid* is on an optimal alignment of A and B. Let $P(A, B)$ denote an optimal alignment of A and B. Partition $P(A, B)$ into two parts immediately after position *imid* of sequence A such that the first part does not end with any insertion gap. Let *jmid* be the largest position of sequence B in the first part. What is the necessary condition on *jmid*? Let A_i^s denote the suffix $a_{i+1}a_{i+2}\ldots a_m$ of sequence A. Notation B_j^s is similarly defined. Then the first part of $P(A, B)$ is an alignment, denoted by $P_1(A_{imid}, B_{jmid})$, of A_{imid} and B_{jmid}, and the second part is an alignment, denoted by $P_2(A_{imid}^s, B_{jmid}^s)$, of A_{imid}^s and B_{jmid}^s.

If $P_1(A_{imid}, B_{jmid})$ ends with a deletion gap and $P_2(A_{imid}^s, B_{jmid}^s)$ begins with a deletion gap, then we have

$$score(P(A, B)) = score(P_1(A_{imid}, B_{jmid})) + score(P_2(A_{imid}^s, B_{jmid}^s)) + q$$

where $score(x)$ is the score of an alignment x. Including the term q on the righthand side ensures that the deletion gap containing both a_{imid} and a_{imid+1} is charged by a

gap open penalty exactly once. Because $P(A, B)$ is an optimal alignment of A and B, $P_1(A_{imid}, B_{jmid})$ has to be a largest-scoring alignment of A_{imid} and B_{jmid} that ends with a deletion gap and $P_2(A^s_{imid}, B^s_{jmid})$ has to be a largest-scoring alignment of A^s_{imid} and B^s_{jmid} that begins with a deletion gap. Define $\bar{D}(i, j)$ to be the maximum score of alignments of A^s_i and B^s_j that begins with a deletion gap. From the definitions of the matrices D and $\bar{D}$, we obtain

$$S(m, n) = D(imid, jmid) + \bar{D}(imid, jmid) + q$$

Because $D(imid, j) + \bar{D}(imid, j) + q$ is the score of an alignment of sequences A and B for each j, $0 \le j \le n$, we have

$$S(m, n) \ge D(imid, j) + \bar{D}(imid, j) + q \quad \text{for each } j,\ 0 \le j \le n$$

Thus $jmid$ is a position j such that $D(imid, j) + \bar{D}(imid, j) + q$ is the maximum. The maximum value is the score of an optimal alignment of A and B.

If $P_1(A_{imid}, B_{jmid})$ does not end with a deletion gap or $P_2(A^s_{imid}, B^s_{jmid})$ does not begin with a deletion gap, then we have

$$score(P(A, B)) = score(P_1(A_{imid}, B_{jmid})) + score(P_2(A^s_{imid}, B^s_{jmid}))$$

Note that $P_1(A_{imid}, B_{jmid})$ cannot end with any insertion gap because of the way $jmid$ is defined. Because $P(A, B)$ is an optimal alignment of A and B, $P_1(A_{imid}, B_{jmid})$ has to be an alignment of A_{imid} and B_{jmid} with the maximum score and $P_2(A^s_{imid}, B^s_{jmid})$ has to be an alignment of A^s_{imid} and B^s_{jmid} with the maximum score. Define $\bar{S}(i, j)$ to be the maximum score of alignments of A^s_i and B^s_j. From the definitions of the matrices S and $\bar{S}$, we obtain

$$S(m, n) = S(imid, jmid) + \bar{S}(imid, jmid)$$

Because $S(imid, j) + \bar{S}(imid, j)$ is the score of an alignment of sequences A and B for each j, $0 \le j \le n$, we have

$$S(m, n) \ge S(imid, j) + \bar{S}(imid, j) \quad \text{for each } j,\ 0 \le j \le n$$

Thus $jmid$ is a position j such that $S(imid, j) + \bar{S}(imid, j)$ is the maximum. The maximum value is the score of an optimal alignment of A and B.

Define df to be

$$df = \max\{D(imid, j) + \bar{D}(imid, j) + q \mid 0 \le j \le n\}$$

Define st to be

$$st = \max\{S(imid, j) + \bar{S}(imid, j) \mid 0 \le j \le n\}$$

Then we have

$$S(m,n) = \max\{df, st\}$$

If $df > st$, then a pair of positions *imid* and *jmid* is on an optimal alignment of sequences A and B, where *jmid* is a position at which the maximum value df is obtained and df is the score of the optimal alignment of A and B. Otherwise, a pair of positions *imid* and *jmid* is on an optimal alignment of sequences A and B, where *jmid* is a position at which the maximum value st is obtained and st is the score of the optimal alignment of A and B.

Define $\bar{I}(i,j)$ to be the maximum score of alignments of A_i^s and B_j^s that begin with an insertion gap. The recurrences for computing the matrices $\bar{S}$, $\bar{D}$, and $\bar{I}$ can be developed similarly as those for the matrices S, D, and I. Here we present the recurrences for $\bar{S}$, $\bar{D}$, and $\bar{I}$ without justification.

$$\bar{S}(m,n) = 0$$

$$\bar{S}(i,n) = \bar{D}(i,n) \quad \text{for } 0 \le i < m$$

$$\bar{S}(m,j) = \bar{I}(m,j) \quad \text{for } 0 \le j < n$$

$$\bar{S}(i,j) = \max\{\bar{S}(i+1,j+1) + \sigma(a_{i+1}, b_{j+1}), \bar{D}(i,j), \bar{I}(i,j)\}$$
$$\text{for } 0 \le i < m \text{ and } 0 \le j < n$$

$$\bar{D}(m,j) = \bar{S}(m,j) - q \quad \text{for } 0 \le j \le n$$

$$\bar{D}(i,n) = \bar{D}(i+1,n) - r \quad \text{for } 0 \le i < m$$

$$\bar{D}(i,j) = \max\{\bar{D}(i+1,j) - r, \bar{S}(i+1,j) - q - r\} \quad \text{for } 0 \le i < m \text{ and } 0 \le j < n$$

$$\bar{I}(i,n) = \bar{S}(i,n) - q \quad \text{for } 0 \le i \le m$$

$$\bar{I}(m,j) = \bar{I}(m,j+1) - r \quad \text{for } 0 \le j < n$$

$$\bar{I}(i,j) = \max\{\bar{I}(i,j+1) - r, \bar{S}(i,j+1) - q - r\} \quad \text{for } 0 \le i < m \text{ and } 0 \le j < n$$

An algorithm for computing an optimal alignment of A and B in linear space consists of the following steps. If m is small enough, compute an optimal alignment of A and B using a traceback procedure. Otherwise, determine a pair of positions *imid* and *jmid* on an optimal alignment of A and B, and recursively compute the portions of the alignment before and after the pair of positions.

The positions *imid* and *jmid* are determined as follows. Set $imid = \lfloor m/2 \rfloor$. Compute the matrices S, D, and I from row 0 to row *imid*, and save $S(imid, j)$ and

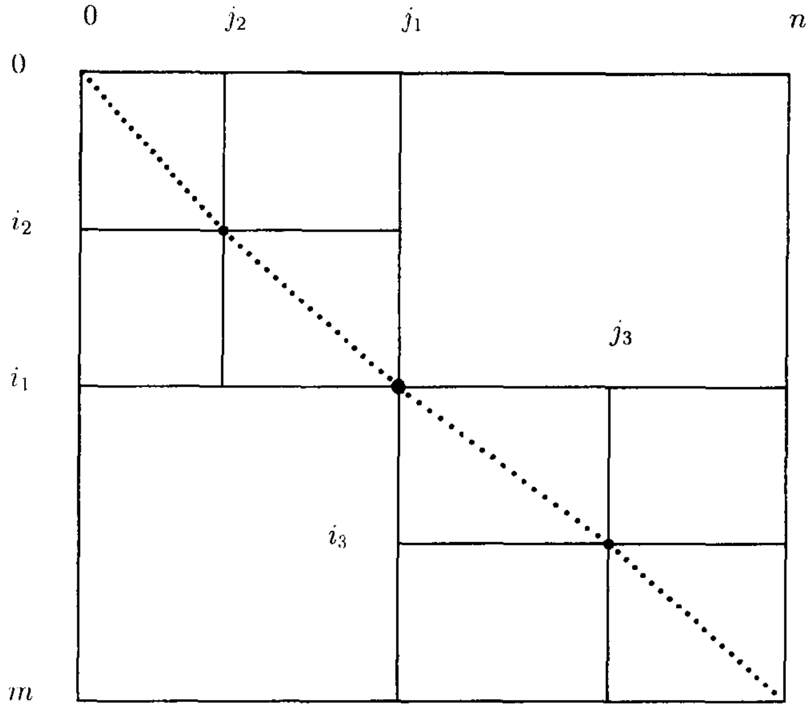


Figure 3.2
Three pairs of positions on an optimal alignment and four sub-subproblems produced by the alignment algorithm after two levels of division. An optimal alignment is indicated by a line of dots. In an initial call to the algorithm, a pair of positions i_1 and j_1 is determined and the original problem is divided into two subproblems. The time required by the non-recursive portion of the algorithm in the initial call is proportional to $m \times n$. The initial call makes two recursive calls, one for each subproblem. In each recursive call, a pair of positions is determined and the subproblem is further divided into two sub-subproblems. The total time required by the non-recursive portion of the algorithm in the two calls is proportional to $(m \times n)/2$.

$D(imid, j)$ for $0 \leq j \leq n$. Compute the matrices $\bar{S}$, $\bar{D}$, and $\bar{I}$ from row m down to row *imid*, and save $\bar{S}(imid, j)$ and $\bar{D}(imid, j)$ for $0 \leq j \leq n$. Let *jd* be a position at which the maximum score *df* is obtained. Let *js* be a position at which the maximum score *st* is obtained. If $df > st$, then set $jmid = jd$. Otherwise, set $jmid = js$. The algorithm is illustrated in figure 3.2.

We first look at the space requirement of the algorithm. Because it requires only a few linear arrays to carry out the computation of the matrices, the algorithm requires space linear in the lengths of sequences. Next we look at the time requirement of the algorithm. Let $t(m,n)$ be the time required by the algorithm to compute an optimal alignment of two sequences of lengths m and n. If m is less than or equal to a constant c_1, then a traceback procedure is used to compute an optimal alignment. Choose a constant c_2 such that

$$t(m,n) \leq c_2(m+n) \quad \text{for } m \leq c_1$$

If m is greater than the constant c_1, then an optimal alignment is computed by finding a pair of positions *imid* and *jmid* on the alignment and computing the portions before and after the pair recursively. The length of A_{imid} is $imid = \lfloor m/2 \rfloor$ and the length of A^s_{imid} is $m - imid = \lceil m/2 \rceil$. Thus the time to compute an optimal alignment of A_{imid} and B_{jmid} is $t(\lfloor m/2 \rfloor, jmid)$ and the time to compute an optimal alignment of A^s_{imid} and B^s_{jmid} is $t(\lceil m/2 \rceil, n - jmid)$. Choose a constant c_3 such that the time on the non-recursive part of the algorithm is at most c_3mn. Thus we have

$$t(m,n) \leq c_3mn + t(\lfloor m/2 \rfloor, jmid) + t(\lceil m/2 \rceil, n - jmid) \quad \text{for } m > c_1$$

It can be proved by induction that

$$t(m,n) \leq 2c_3mn + 2c_2(m+n)$$

This means that the algorithm takes time in proportion to the product of the sequence lengths.

3.2.4 Performing Computation in a Band

One approximation for reducing the time of the global alignment algorithm is to restrict the computation to a band of diagonals in each matrix (Sankoff and Kruskal 1983; Pearson and Lipman 1988). A diagonal k of a matrix consists of those entries (i,j) with $j - i = k$. A band from diagonals ld to hd consists of those entries with $ld \leq j - i \leq hd$. If sequences A and B are very similar, it is likely that an optimal alignment of A and B is completely within a narrow band of diagonals. To carry out the computation in a band of diagonals, each entry outside the band is given a value of negative infinity and each entry inside the band is computed according to the recurrences. Note that any band that covers an optimal alignment of A and B has to contain entries $(0,0)$ and (m,n). Later we describe a fast method to estimate the width of a band so that it is likely to cover an optimal alignment. However, the method does not guarantee that the band always covers an optimal alignment. Chao et al. (1992) developed an efficient algorithm for computing an alignment in a band. Others proposed a few computational techniques to compute an optimal alignment in a band or a small matrix area (Fickett 1984; Ukkonen 1985; Spouge 1991).

3.3 Local Alignment

The global alignment algorithm described above is intended for sequences that are similar over their entire lengths. However, there are situations where two sequences are not globally similar, but contain similar regions. For instance, genomic sequences

from distantly related organisms contain short similar exons, but long different introns and intergenic regions. A local alignment algorithm should be used to find similar regions between two sequences. A local alignment between two sequences A and B is an alignment of a region of A and a region of B. An optimal local alignment between A and B is a local alignment with the maximum score.

An algorithm for computing an optimal local alignment between A and B is developed using dynamic programming. Define $LS(i, j)$ (L for local) to be the maximum score of local alignments ending at positions i and j of A and B. Similarly, define $LD(i, j)$ for alignments that end with a deletion gap and $LI(i, j)$ for alignments that end with an insertion gap. The recurrences for computing the matrices LS, LD, and LI are developed in a similar way as those for the matrices in the global alignment algorithm.

$$LS(i, j) = 0 \quad \text{for } i = 0 \text{ or } j = 0$$

$$LS(i, j) = \max\{0, LS(i-1, j-1) + \sigma(a_i, b_j), LD(i, j), LI(i, j)\} \quad \text{for } i > 0 \text{ and } j > 0$$

$$LD(0, j) = -q \quad \text{for } j \geq 0$$

$$LD(i, j) = \max\{LD(i-1, j) - r, LS(i-1, j) - q - r\} \quad \text{for } i > 0 \text{ and } j > 0$$

$$LI(i, 0) = -q \quad \text{for } i \geq 0$$

$$LI(i, j) = \max\{LI(i, j-1) - r, LS(i, j-1) - q - r\} \quad \text{for } i > 0 \text{ and } j > 0$$

The zero in the recurrence for LS is the score of the empty local alignment, an alignment of two regions of length 0. The zero in the recurrence serves two purposes. First, an optimal local alignment can start at any positions i and j in sequences A and B. There is no penalty for not including, in the optimal local alignment, the initial regions of A and B before positions i and j. Second, any local alignment of a negative score is ignored because it cannot be an initial portion of any optimal local alignment. The justification for the recurrences is similar to that for the recurrences in the global alignment algorithm and is omitted.

An entry (ie, je) with the maximum value in the matrix LS is the end point of an optimal local alignment between A and B. The optimal local alignment can be found by a traceback procedure starting at the entry (ie, je), which requires quadratic space. This algorithm is the result of Smith and Waterman (1981) and Gotoh (1982). Selecting an entry with the maximum value serves similar purposes to terminal regions of A and B as including the zero in the recurrence to initial regions of A and B. Those two features in the local alignment algorithm are responsible for the generation of an optimal local alignment, instead of an optimal global alignment.

Alternatively, an optimal local alignment is computed in linear space by first determining its start point (is, js) and then applying the linear space global alignment procedure to sequences $a_{is}a_{is+1} \dots a_{ie}$ and $b_{js}b_{js+1} \dots b_{je}$. The end point (ie, je) is an entry with the maximum value in the matrix LS. The start point (is, js) is obtained by computing the matrices $\bar{S}$, $\bar{D}$, and $\bar{I}$ with respect to sequences A_{ie} and B_{je}. Then (is, js) is an entry with the maximum value in the matrix $\bar{S}$.

Waterman and Eggert (1987) generalized the algorithm to compute k best local alignments between two sequences. Two local alignments are independent if they share no substitution from a common pair of positions in the sequences. A first best local alignment is an optimal local alignment between the two sequences. A second best local alignment is a largest-scoring local alignment that is independent of the first best local alignment. A third best local alignment is a largest-scoring local alignment that is independent of the first and second best local alignments. Other best local alignments are similarly defined. Huang and Miller (1991) developed a space-efficient algorithm, SIM. The SIM algorithm computes k best local alignments between two sequences in linear space.

3.4 A Fast Algorithm

The sequence alignment algorithms described in the previous sections take time in proportion to the product of sequence lengths. Thus it is impractical to use those alignment algorithms to compare very long sequences. Fast approximation algorithms are required to compare long sequences. Below we describe a fast algorithm to identify similar regions between two sequences and to produce an alignment for each pair of similar regions.

The fast algorithm consists of three major steps. In step 1, high-scoring segment pairs between the two sequences are computed. A segment pair is an alignment without any gaps. Segment pairs of scores greater than a cutoff are saved for the next step. In step 2, high-scoring chains of segment pairs are computed using dynamic programming and chains that begin with the same segment pair are grouped together. The score of a chain group is the maximum score of chains in the group. In step 3, for each chain group of score greater than a cutoff, a chain with the maximum score in the group is selected. The two sequence regions involved in the chain and a band of diagonals that covers all segment pairs in the chain are determined. Then the linear-space global alignment algorithm is applied to the two regions to compute a largest-scoring alignment of the regions over the band of diagonals. We define chains of segment pairs and describe computation of chains of segment pairs in detail below.

3.4.1 Chains of Segment Pairs

A segment pair between sequences A and B is a gap-free alignment of two segments of A and B. The score of a segment pair is the sum of scores of each match and mismatch in the segment pair. For a segment pair s, let $astart(s)$ and $aend(s)$ denote the starting and ending positions of the segment in sequence A, let $bstart(s)$ and $bend(s)$ denote the starting and ending positions of the segment in sequence B, and let $score(s)$ denote the score of s. The first antidiagonal of a segment pair s is defined to be $antis(s) = astart(s) + bstart(s)$, and the last antidiagonal of s is defined to be $antid(s) = aend(s) + bend(s)$.

A chain of segment pairs is a list of segment pairs in increasing order of their last antidiagonals such that each segment pair is not far from its predecessor and adjacent segment pairs do not have a large overlap. Specifically, any two adjacent segment pairs s and s' in the list satisfy the requirement

$$antis(s') - antid(s) < d_1$$

$$astart(s') - aend(s) > -d_2$$

$$bstart(s') - bend(s) > -d_2$$

for some nonnegative integers d_1 and d_2. Let $close(s, s')$ denote the condition given above. A chain of segment pairs is used as an approximation of a local alignment between sequences A and B with the segment pairs being ungapped portions of the alignment. Note that the use of the d_1 cutoff permits efficient computation of high-scoring chains.

A linear gap penalty is charged for the regions between two adjacent segment pairs. For some nonnegative integers q and r, the penalty for connecting two segment pairs s and s' is

$$gap(s, s') = q + r \times [l(astart(s') - aend(s)) + l(bstart(s') - bend(s))]$$

where $l(x) = x$ if $x > 0$ and 0 otherwise. For two adjacent segment pairs s and s' in a chain, define $tscore(s, s')$ to be the score of the longest portion of s' that has no overlap with s. The score of a chain c of segment pairs $s_1, s_2, \ldots, s_k$ is defined to be

$$score(c) = score(s_1) + \sum_{i=2}^{k} [tscore(s_{i-1}, s_i) - gap(s_{i-1}, s_i)]$$

To ensure that each segment pair contributes to the chain, we require that for any two adjacent segment pairs s and s' in the chain, $tscore(s, s')$ be greater than a cutoff

ic. Two segment pairs s and s' are identical if $astart(s) = astart(s')$ and $bstart(s) = bstart(s')$. Two chains of segment pairs are non-intersecting if they don't have any common segment pair (Chao and Miller 1995).

3.4.2 Fast Computation of Chains of Segment Pairs

Segment pairs of scores greater than a cutoff between sequences A and B are approximately computed using a hashing technique, as follows. Assume that $m \leq n$. A lookup table is constructed for sequence A such that for each word of length w, the table provides the positions of each occurrence of the word in sequence A. The word length w is chosen such that the size of the lookup table is close to the length m of A. For each position p of sequence B, a word of length w beginning at position p of B is considered as follows. The lookup table is used to locate each occurrence of the word in sequence A. Each exact match of length w is extended in both directions until the score drops below the maximum score by at least d_3 units (Altschul et al. 1990). If a word match is contained in a segment pair already considered, the match is not extended. The segment pair of the maximum score found during the extension is saved if the score is greater than the cutoff.

After the computation of segment pairs between A and B, non-intersecting chains of segment pairs with scores greater than a cutoff f are computed. Let $s_1, s_2, \ldots, s_k$ be a list of all the segment pairs in increasing order of their last antidiagonals. Let $Q(s_i)$ be the maximum score of chains ending with segment pair s_i. The matrix Q is computed using dynamic programming (Wilbur and Lipman 1983; Pearson and Lipman 1988; Chao and Miller 1995; Huang 1996).

$$Q(s_1) = score(s_1)$$

$$Q(s_i) = \max\{score(s_i), Q(s_j) + tscore(s_j, s_i) - gap(s_j, s_i) \mid 1 \leq j < i, close(s_j, s_i), \text{and } tscore(s_j, s_i) > ic\} \quad \text{for } i > 1$$

For segment pairs s_j and s_i with $j < i$, if the overlap cutoffs d_1 and d_2 are violated or the score of the nonoverlapping portion of s_i is not large enough, then s_j is excluded from consideration as an immediate predecessor to s_i in any chain. To compute $Q(s_i)$, it suffices to use each s_j in decreasing value of j such that $antid(s_j) > antis(s_i) - d_1$. To compute $tscore(s_j, s_i)$ efficiently for each s_j, an array R of size d_2 is computed for s_i before $Q(s_i)$, where for $0 \leq t < d_2$, $R(t)$ is the sum of the scores of the first $t+1$ aligned pairs in s_i if there are at least $t+1$ aligned pairs in s_i and $score(s_i)$ otherwise. Let $aover(s_j, s_i)$ denote $astart(s_i) - aend(s_j)$ and let $bover(s_j, s_i)$ denote $bstart(s_i) - bend(s_j)$. Then for each s_j, if $aover(s_j, s_i) > 0$ and $bover(s_j, s_i) > 0$, then $tscore(s_j, s_i)$ is

equal to $score(s_i)$. Otherwise, we have

$$tscore(s_j, s_i) = score(s_i) - R(\max\{-aover(s_j, s_i), -bover(s_j, s_i)\})$$

Largest-scoring chains of segment pairs are partitioned into equivalence classes by the starting segment pair of the chains (Huang and Miller 1991; Chao and Miller 1995). Two chains are in the same class if and only if they begin with the same segment pair. The score of an equivalence class is the maximum score of chains in the class.

Chain classes of scores greater than f can be easily computed along with the matrix Q as follows (Huang and Miller 1991; Chao and Miller 1995). Let $K(s_i)$ be the first segment pair of a largest-scoring chain ending with segment pair s_i. For each segment pair s_i, $K(s_i)$ is initialized to s_i. When $Q(s_i)$ is set to $Q(s_j) + tscore(s_j, s_i) - gap(s_j, s_i)$, $K(s_i)$ is set to $K(s_j)$. For an equivalence class c, let $start(c)$ be the starting segment pair for the class, let $end(c)$ be the ending segment pair of a largest-scoring chain in the class, and let $score(c)$ be the score of the class. Thus we have $Q(end(c)) = score(c)$. The equivalence classes of scores greater than f are saved. After $Q(s_i)$ and $K(s_i)$ are computed, we perform one of the two tasks below if $Q(s_i)$ is greater than f. If there is an equivalence class c with $start(c) = K(s_i)$, set $end(c)$ to s_i and $score(c)$ to $Q(s_i)$ if $score(c) < Q(s_i)$. If there is no equivalence class c with $start(c) = K(s_i)$, create a new class c with $start(c) = K(s_i)$, $end(c) = s_i$, and $score(c) = Q(s_i)$. After the computation of the equivalence classes is completed, for each saved equivalence class, a largest-scoring chain in the class is obtained by a traceback technique. These largest-scoring chains are nonintersecting. To see this, if two chains were intersecting, that is, they had a common segment pair s, then the two chains would begin with the same segment pair $K(s)$ and hence would belong to the same equivalence class. This contradicts the fact that the two chains are from different equivalence classes.

3.5 An Algorithm for Comparing Two Sets of Sequences

We consider a general problem of comparing every sequence in one set with every sequence in the other set. The goal is to find pairs of sequences with similar regions between the two sets and to report those similar regions. If one set is a large database of sequences and the other set is a set of query sequences, then the problem is a database searching problem.

We develop an efficient algorithm for this general problem as follows. All sequences in the smaller set are concatenated to form a composite string with a new character

inserted at every sequence boundary (Huang and Madan 1999). One lookup table is made for the composite string. For each sequence in the larger set, the fast algorithm in the previous section is used to compare the sequence with the composite string through the lookup table. Special care is taken to ensure that no word match is extended beyond any sequence boundary indicated by the new character in the composite string and that only segment pairs from the same sequence in the composite string can be combined into chains. Note that the construction of one lookup table for the composite string enables us to find directly pairs of sequences with similar regions without going through every pair of sequences from the two sets, many of which may not contain similar regions.

3.6 Applications

We present four applications of sequence alignment programs to analysis of DNA and protein sequences. First, we give an example of using a global alignment program to compare homologous human and mouse protein sequences. Second, we look at a case of using a global alignment program to compare syntenic human and mouse genomic DNA sequences. Third, we provide an example of using a fast comparison program and a rigorous alignment program to identify the exon-intron boundaries of genes in a genomic DNA sequence. Fourth, we give an instance of using a fast comparison program to determine the similarity relationships between two large sets of sequences. All the applications were performed on a Sun Ultra 5 workstation with 128 Mb of main memory.

3.6.1 Comparison of Two Protein Sequences

A novel gene named *Usp29* was recently found in a region of mouse chromosome 7 and a homologous region of human chromosome 19 (Kim et al. 2000). The cDNA sequence of mouse gene *Usp29* encodes a protein of 869 amino acids (GenBank accession no. AF229257). Because the sequence of the mouse protein is similar to the sequences of yeast and nematode proteins from the type-2 family of ubiquitin carboxyl-terminal hydrolases, the mouse protein is likely to function as a ubiquitin carboxyl-terminal hydrolase and is therefore named *Usp29* (ubiquitin-specific processing protease 29). (Ubiquitin carboxyl-terminal hydrolase is also known as ubiquitin-specific processing protease.) Proteins in the type-2 family contain two conserved domains named the cys box and the his box, which define the active sites of those proteins. The cDNA sequence of human gene *Usp29* encodes a protein of 922 amino acids (GenBank accession no. AF229438). Two questions could be asked about the mouse and human proteins. What is the level of overall sequence conservation between

the mouse and human proteins? Are the two conserved domains that are unique to the type-2 family highly conserved between the mouse and human proteins?

The two questions were addressed by computing an optimal alignment of the mouse and human protein sequences with a program named GAP (global alignment program). The GAP program computes an optimal global alignment of two sequences in quadratic time and linear space, where terminal gaps are not penalized and long internal gaps in the shorter sequence are given a constant penalty (Huang 1994). An alignment of the two mouse and human *Usp29* sequences was produced by GAP with the following values for its parameters: BLOSUM62 for substitution matrix, 15 for gap open penalty, and 2 for gap extension penalty. The running time of GAP was less than a second. The alignment showed that the two sequences have a low identity of 41 percent, well below an average identity of 85 percent between human and mouse protein sequences. However, the two conserved domains are highly conserved between the mouse and human proteins (figure 3.3). The high level of sequence conservation between the two domains of the mouse and human proteins also suggests that the mouse and human proteins belong to the type-2 family.

3.6.2 Comparison of Two Genomic Sequences

We look at an example of comparing two large genomic sequences from syntenic regions of the human and mouse genomes. The number, order, and orientation of genes in syntenic regions of two different species are conserved between the two species. The 223-kb human genomic sequence (GenBank accession no. U47924) is from a gene-rich cluster at the *CD4* locus on human chromosome 12p13 (Ansari-Lari et al. 1996). The 227-kb mouse genomic sequence (GenBank accession no. AC002397) is from the syntenic region on mouse chromosome 6 (Ansari-Lari et al. 1998). The two *CD4* sequences were previously compared with a modified version of SIM program by Ansari-Lari et al. (1998). In this application, we show that coding regions in the two *CD4* sequences can be identified by computing a global alignment of the two sequences.

A program named GAP3 was used to compare the *CD4* genomic sequences. The GAP3 program computes an optimal global alignment of two sequences in quadratic time and linear space, where long, different regions in the two sequences are given a constant penalty (Huang, unpublished results). To align the two *CD4* sequences on the basis of coding regions, instead of repeat elements, the repeat elements in the *CD4* genomic sequences were masked by RepeatMasker (Smit and Green 1996) and the masked versions of the sequences were used by GAP3 for alignment. The GAP3 program produced a large alignment of the two sequences, which contains 46,019 base matches (20 percent). Many of the matching regions on the alignment corre-

```
Upper Sequence: Mouse USP29 protein sequence   Length: 869
Lower Sequence: Human USP29 protein sequence   Length: 922

       ... A portion of 250 aa is removed ...

250     .    :    .    :    .    :    .    :    .    :
249 TILPIATCSDDRDVSIFGLEIITHNG     VQSLPDPYLNQLKREGFPN
    | |   |         - ||     |-----    |     |    ||||
241 TVLATQTLNAKNGLTS PLEPEHSQGDPRCNKAQVPLDSHSQQLQQGFPN
                                                   ***

300     .    :    .    :    .    :    .    :    .    :
294 LGNTCYMNSILQSVFGIPTFAKDLLTQGIPWEKVSYDDLIMPLSQLLVLK
    ||||||||  ||| | || || |||||| |||      ||| | | | ||
290 LGNTCYMNAVLQSLFAIPSFADDLLTQGVPWEYIPFEALIMTLTQLLALK
    ********

350     .    :    .    :    .    :    .    :    .    :
344 DIRDVEIKGELLTSVKKSISTVADTFSGNEQNDAHEFLSLCLDQLKLNME
    |     || |||  ||| || ||  |||| |||||||   ||||||  ||
340 DFCSTKIKRELLGNVKKVISAVAEIFSGNMQNDAHEFLGQCLDQLKEDME

       ... A portion of 350 aa is removed ...

750     .    :    .    :    .    :    .    :    .    :
705 HGSRIKGLFL    PASLASVSSQEDPEKDLSRSPELQ   EDDPHSFAF
        |   ||----|        ||  |  |  | || --- |  |  |
738 IEESIIDEFLQQAPPPGVRKLDAQEHTEETLNQSTELRLQKADLNHLGAL

800     .    :    .    :    .    :    .    :    .    :
748 GSDD                   SKDGEMGDDLQNYRLVSVVSHFGSSPN
     || -------------------     ||| || ||| ||||| |||||
788 GSDNPGNKNILDAENTRGEAKELTRNVKMGDPLQAYRLISVVSHIGSSPN
                                       ***************

850     .    :    .    :    .    :    .    :    .    :
779 SGHYVSDVYDFQKQAWLLYSDVQVFESSDPSIQENRLNSGYIFFYMHNEI
    |||| |||||||||||  | |  | | |    || || |||||||||| |
838 SGHYISDVYDFQKQAWFTYNDLCVSEISETKMQEARLHSGYIFFYMHNGI
    ****

900     .    :    .    :    .    :    .    :
829 FEELLKKASECKVLSTSKEEKRDIDYFSTLLNGLTYILEEF
    ||||| ||      ||         |    |    ------
888 FEELLRKAENSRLPSTQAGVIPQGEYEGDSLYRPA
```

Figure 3.3
Portions of an alignment of mouse and human *Usp29* protein sequences. Two conserved domains are indicated by asterisks.

```
Upper Sequence: Mouse CD4 genomic sequence        Length: 227538 bp
Lower Sequence: Human CD4 genomic sequence        Length: 222930 bp

142500     .    :    .    :    .    :    .    :    .    :    .    :
 84882          TCTCTGCCCCACAGATAATGTGAAAGACTGGAGCAAGGTGGTCCTGGCCTA
       ---------|||||||||  ||||||| ||||| |||||||||||||| |||||||||||
 81675 CCATCTGTATCTCTGCCCTGCAGATAACGTGAAGGACTGGAGCAAGGTCGTCCTGGCCTA
                          ++++****

142560     .    :    .    :    .    :    .    :    .    :    .    :
 84933 TGAACCTGTGTGGGCCATTGGGACTGGCAAGACGGCAACCCCTCAGCAGGTAAC
       ||| ||||||||||||||||| ||||||||||| ||||| || || ||||||||------
 81735 TGAGCCTGTGTGGGCCATTGGTACTGGCAAGACTGCAACACCCCAACAGGTAACCGGGCC
                                                    ****++++

           ~~~A gap of 480 bp is removed ~~~

143100     .    :    .    :    .    :    .    :    .    :    .    :
 85209 AATGATAGATAGGAAGGAGCTTACTTAGTAGCTCTGCCTCCTCCTTCTAGGCACAGGAAG
       --------------------------------------------|||||||| |||||||
 82053                                             TTCTAGGCCCAGGAAG
                                                     ++++****

143160     .    :    .    :    .    :    .    :    .    :    .    :
 85269 TACACGAGAAGCTCCGGGGATGGCTGAAATCCAATGTCAATGATGGGGTGGCTCAGAGCA
       |||||||||||||||| ||||||||||| ||||| |||  ||||| ||||||||||||||
 82069 TACACGAGAAGCTCCGAGGATGGCTGAAGTCCAACGTCTCTGATGCGGTGGCTCAGAGCA

143220     .    :    .    :    .    :    .    :    .    :    .    :
 85329 CCCGGATCATTTATGGAGGTGAGTGGCTTTGGCTC
       |||| ||||||||||||||||||||||||||| ||-------------------------
 82129 CCCGTATCATTTATGGAGGTGAGTGGCTTTGGTTCCCGGCTGAGGTGGAGTGGGCTGAGG
                     ****++++
```

Figure 3.4
Portions of a large alignment of mouse and human CD4 genomic sequences. Two mouse exons are correctly aligned with two human exons with respect to exon-intron boundaries. The four exon-intron boundaries are indicated by asterisks under exon bases and plus signs under intron bases.

spond to exons of the sequences. Portions of the alignment corresponding to two exons are shown in figure 3.4. The following values were used for the parameters of GAP3: 10 for match score, −15 for mismatch score, 60 for gap open penalty, and 3 for gap extension penalty. The computation took 4 hours and 39 minutes and 6.7 Mb of main memory.

3.6.3 Identification of Exon-Intron Boundaries

In this application, we demonstrate that sequence alignment programs are useful for finding the exon-intron boundaries of a gene in a genomic sequence if the cDNA sequence of the gene is known. The genomic sequence used in this example is the *CD4* mouse genomic sequence from the last subsection. The mouse genomic sequence

contains a gene whose cDNA sequence had been determined eight years earlier. The cDNA sequence (GenBank accession no. NM_013509) encodes a protein of 434 amino acids, which functions as a gamma enolase. The mouse genomic sequence was compared with the cDNA sequence by a software tool named AAT (analysis and annotation tool) (Huang et al. 1997).

The AAT tool contains a fast database search program named DDS (DNA-DNA search) and a rigorous alignment program named GAP2 for comparing a genomic sequence with a database of cDNA sequences. The DDS program quickly computes high-scoring chains of segment pairs between the genomic sequence and the database of cDNA sequences. Every high-scoring chain indicates that a region of the genomic sequence is similar to a cDNA sequence. For each pair of a genomic region and a cDNA sequence, the GAP2 program computes an optimal alignment of the genomic region and the cDNA sequence. The GAP2 program is an improvement to the GAP program, where dinucleotides AG and GT are used by GAP2 to identify exon-intron boundaries.

On the CD4 mouse genomic sequence and the cDNA sequence, DDS reported a high-scoring chain between a region of the CD4 sequence from bases 129,471 to 137,445 and the cDNA sequence. The GAP2 program produced a 8,424-bp alignment of the genomic region and the cDNA sequence, where 11 exons of the genomic region are aligned with portions of the cDNA sequence. Portions of the alignment are shown in figure 3.5. The exon-intron boundaries identified by GAP2 in the genomic region are exactly identical to those reported in the GenBank entry of the CD4 mouse sequence. The DDS program took less than a second and the GAP2 program took 22 seconds on the data. The default values were used for the parameters of DDS and GAP2. Note that in this application, the database just contains one cDNA sequence. In a real situation, the database contains all cDNA sequences that have been produced.

3.6.4 Comparison of Two Sets of Sequences

We describe an application of a program to comparison of two large sets of sequences. We developed a version, named DDS.BTAB, of the DDS program (Huang et al. 1997) for comparing two sets of sequences. The DDS.BTAB program quickly computes high-scoring chains between sequences in one set and sequences in the other set. The DDS.BTAB program was applied to comparison of two sets of sequences produced by two DNA sequence assembly programs. The two assembly programs were used to assemble the same set of raw DNA fragments into long sequences. One program produced a set of 47 sequences of a total of 1.9 megabases; the other program produced a set of 623 sequences of a total of 2.2 megabases. Obviously, the

```
Upper Sequence: Mouse ENO2 genomic region from bases 129471 tc 137445
Lower Sequence: Mouse ENO2 cDNA sequence                   Length: 2341

    5940        .    :    .    :    .    :    .    :    .    :    .    :
  135012 TTCGGGGCAGGTCAGTGCTGGTGTACGTACCCTGCAGGTGCAAGCTGGCCCAGGAGAATG
         -------------------------------------||||||||||||||||||||||||
    1181                                      GTGCAAGCTGGCCCAGGAGAATG

    6000        .    :    .    :    .    :    .    :    .    :    .    :
  135072 GCTGGGGGGTTATGGTGAGCCATCGCTCTGGAGAAACGGAGGACACGTTCATTGCTGATC
         |||||||||||||||||||||||||||||||||||||||||||||||| ||||||||||||
    1204 GCTGGGGGGTTATGGTGAGCCATCGCTCTGGAGAAACGGAGGACACGTTCATTGCTGATC

Exon 9 is from bases 135049 to 135157

    6060        .    :    .    :    .    :    .    :    .    :    .    :
  135132 TTGTCGTCGGACTGTGTACAGGCCAGGTGAGTAGAGCCAGTCTGTGGGATTGGGAGAGGT
         ||||||||||||||||||||||||||----------------------------------
    1264 TTGTCGTCGGACTGTGTACAGGCCAG


            ~~~A gap of 900 bp is removed ~~~

    7020        .    :    .    :    .    :    .    :    .    :    .    :
  136092 GGAATGTTTCAGATCAAGACTGGCGCCCCATGCAGATCTGAACGTCTGGCGAAGTACAAC
         ------------||||||||||||||||||||||||||||||||||||||||||||||||
    1290             ATCAAGACTGGCGCCCCATGCAGATCTGAACGTCTGGCGAAGTACAAC

Exon 10 is from bases 136104 to 136162

    7080        .    :    .    :    .    :    .    :    .    :    .    :
  136152 CAGCTCATGAGGTGAGGAGGGTCCCCGAAGAACAAGAACCCGAAACCCGGGGTCTAAACG
         |||||||||||-------------------------------------------------
    1338 CAGCTCATGAG
```

Figure 3.5
Portions of an alignment of a genomic region and a cDNA sequence. The cDNA sequence is correctly aligned with the genomic region with respect to exon-intron boundaries. The 5′ and 3′ coordinates of exons 9 and 10 are shown.

results from the two assembly programs were quite different. We wanted to know the major differences between the two assembly results by finding major similarities between the two sets of sequences. The DDS.BTAB program was used to compute major similarities between the two sets of sequences.

The DDS.BTAB program produced 286 chains of scores greater than 2,000, where a match was given a score of 2, a mismatch a score of -3, a gap was penalized with a gap open penalty of 10, and a gap extension penalty of 1, and segment pairs of scores greater than 80 were used. A high value of 2,000 was used for the chain score cutoff in order for DDS.BTAB to report only significant matches. The computation took 68 seconds. The word length used in this run was 11. Those major matches between the

two sets of sequences allowed us to figure out the relationships between the two sets of sequences.

3.7 Future Developments

We suggest two directions for developments of new and improved sequence comparison methods. One direction is to improve existing methods so that they can distinguish distantly related sequences from unrelated sequences. If two related protein sequences have an identity of 30 percent or higher, then existing methods can determine that the two sequences are related. On the other hand, if two protein sequences have an identity of 25 percent, then existing methods cannot determine if the two sequences are related or not. The other direction is to develop new methods for comparing two large genomes, such as the human and mouse genomes. The immediate objectives of the genome comparison are to identify conserved coding regions and regulatory elements between the two genomes. Exons are often conserved between the human and mouse genomes, whereas introns and intergenic regions are often divergent. The new methods must be efficient enough to handle huge sequences and have new features to address various issues.

Acknowledgments

I would like to thank the reviewers and editors for many helpful suggestions on the presentation of this chapter.

References

Altschul, S. F., Gish, W., Miller, W., Myers, E. W., and Lipman, D. J. (1990). Basic local alignment search tool. *J. Mol. Biol.* 215: 403–410.

Ansari-Lari, M. A., Muzny, D. M., Lu, J., Lu, F., Lilley, C. E., Spanos, S., Malley, T., and Gibbs, R. A. (1996). A gene-rich cluster between the CD4 and triosephosphate isomerase genes at human chromosome 12p13. *Genome Res.* 6: 314–326.

Ansari-Lari, M. A., Oeltjen, J. C., Schwartz, S., Zhang, Z., Muzny, D. M., Lu, J., Gorrell, J. H., Chinault, A. C., Belmont, J. W., Miller, W., and Gibbs, R. A. (1998). Comparative sequence analysis of a gene-rich cluster at human chromosome 12p13 and its syntenic region in mouse chromosome 6. *Genome Res.* 8: 29–40.

Attwood, T. K. (2000). The babel of bioinformatics. *Science* 290: 471–473.

Chao, K.-M., and Miller, W. (1995). Linear-space algorithms that build local alignments from fragments. *Algorithmica* 13: 106–134.

Chao, K.-M., Pearson, W. R., and Miller, W. (1992). Aligning two sequences within a specified diagonal band. *Comput. Appl. Biosci.* 8: 481–487.

Dayhoff, M. O., Schwartz, R. M., and Orcutt, B. C. (1978). A model of evolutionary change in proteins. In Atlas of Protein Sequence and Structure. Dayhoff, M. O. ed., vol. 5, suppl. 3, 345–358. Washington, DC: National Biomedical Research Foundation.

Fickett, J. W. (1984). Fast optimal alignment. *Nucl. Acids Res.* 12: 175–180.

Gotoh, O. (1982). An improved algorithm for matching biological sequences. *J. Mol. Biol.* 162: 705–708.

Henikoff, S., and Henikoff, J. G. (1992). Amino acid substitution matrices from protein blocks. *Proc. Natl. Acad. Sci. USA* 89: 10915–10919.

Hirschberg, D. S. (1975). A linear space algorithm for computing maximal common subsequences. *Commun. Assoc. Comput. Mach.* 18: 341–343.

Huang, X. (1994). On global sequence alignment. *Comput. Appl. Biosci.* 10: 227–235.

Huang, X. (1996). Fast comparison of a DNA sequence with a protein sequence database. *Microbial & Comparative Genomics* 1: 281–291.

Huang, X., Adams, M. D., Zhou, H., and Kerlavage, A. R. (1997). A tool for analyzing and annotating genomic sequences. *Genomics* 46: 37–45.

Huang, X., and Madan, A. (1999). CAP3: A DNA sequence assembly program. *Genome Res.* 9: 868–877.

Huang, X., and Miller, W. (1991). A time-efficient, linear-space local similarity algorithm. *Adv. Appl. Math.* 12: 337–357.

Kim, J., Noskov, V. N., Lu, X., Bergmann, A., Ren, X., Warth, T., Richardson, P., Kouprina, N., and Stubbs, L. (2000). Discovery of a novel, paternally expressed ubiquitin-specific processing protease gene through comparative analysis of an imprinted region of mouse chromosome 7 and human chromosome 19q13.4. *Genome Res.* 10: 1138–1147.

Makalowski, W., Zhang, J., and Boguski, M. S. (1996). Comparative analysis of 1196 orthologous mouse and human full-length mRNA and protein sequences. *Genome Res.* 6: 846–857.

Myers, E. W., and Miller, W. (1988). Optimal alignments in linear space. *Comput. Applic. Biosci.* 4: 11–17.

Needleman, S. B., and Wunsch, C. D. (1970). A general method applicable to the search for similarities in the amino acid sequences of two proteins. *J. Mol. Biol.* 48: 443–453.

Pearson, W. R., and Lipman, D. (1988). Improved tools for biological sequence comparison. *Proc. Natl. Acad. Sci. USA* 85: 2444–2448.

Sankoff, D., and Kruskal, J. B. eds. (1983). *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparisons.* Reading, Mass: Addison-Wesley.

Sellers, P. H. (1974). On the theory and computation of evolutionary distances. *SIAM J. Appl. Math.* 26: 787–793.

Smit, A. F. A., and Green, P. 1996. http://ftp.genome.washington.edu/cgi-bin/RepeatMasker.

Smith, T. F., and Waterman, M. S. (1981). Identification of common molecular subsequences. *J. Mol. Biol.* 147: 195–197.

Spouge, J. L. (1991). Fast optimal alignment. *Comput. Applic. Biosci.* 7: 1–7.

Ukkonen, E. (1985). Algorithms for approximate string matching. *Information and Control* 64: 100–118.

Wagner R. A., and Fischer, M. J. (1974). The string-to-string correction problem. *J. Assoc. Comput. Mach.* 21: 168–173.

Waterman, M. S., and Eggert, M. (1987). A new algorithm for best subsequence alignments with application to tRNA-rRNA comparisons. *J. Mol. Biol.* 197: 723–728.

Waterman, M. S., Smith, T. F., and Beyer, W. A. (1976). Some biological sequence metrics. *Adv. Math.* 20: 367–387.

Wilbur, W. J., and Lipman, D. J. (1983). Rapid similarity searches of nucleic acid and protein data banks. *Proc. Natl. Acad. Sci. USA* 80: 726–730.

4 Algorithmic Methods for Multiple Sequence Alignment

Tao Jiang and Lusheng Wang

4.1 Introduction

Multiple sequence alignment is a fundamental and challenging problem in computational molecular biology (Altschul and Lipman 1989; Carrillo and Lipman 1988; Gusfield 1993, 1997; Sankoff and Kruskal 1983; Waterman 1995). Algorithms for multiple sequence alignment are routinely used to find conserved regions in biomolecular sequences, to construct family and superfamily representations of sequences, and to reveal evolutionary histories of species (or genes). Conserved subregions in DNA/protein sequences may represent important functions or regulatory elements. The profile or consensus sequence obtained from a multiple alignment can be used to characterize a family or superfamily of species. Multiple sequence alignment is also closely related to phylogenetic analysis. For example, most phylogeny reconstruction algorithms use multiple sequence alignments as their input. Moreover, some versions of multiple sequence alignment, such as *tree alignment*, can be used directly to measure the goodness of candidate trees (Wang et al. 2001). Along with the fantastic advances in worldwide sequencing projects, efficient methods for multiple sequence alignment are becoming ever more important for understanding the sequences that are being produced every day.

From a mathematical point of view, multiple sequence alignment is a a natural extension of pairwise sequence alignment (see chapter 3). Computer programs for multiple sequence alignment are becoming critical tools for biological sequence analysis, and can help extract and represent biologically important commonalities (conserved motifs, conserved characters in DNA or protein, common secondary or tertiary structures, etc.) from a set of sequences. Biological commonalities may be very non-obvious and hard to detect, especially when two sequences are being compared. However, they may become more clear when a set of related sequences are being compared. Below, we include two examples of multiple sequence alignment. The first example is given in figure 4.1 (Gusfield 1997; McLure et al. 1994). The abbreviations on the left indicate the organisms that the globin sequences are from. Because of the lengths of the sequences, the multiple alignment is folded into three sections. Columns in the alignment containing a high concentration of similar residues in regions of known secondary structure are marked with a "v," and columns with identical residues are marked with a star. Two residues are considered similar if they belong to the same class of the following partition: (F,Y), (M,L,I,V), (A,G), (T,S), (Q,N), (K,R), and (E,D).

```
                                                  *vvvvv*

HUMA       VLSPADKTNVKAAWGKVGAHAGEYGAEALERMFLSFPTTKTYFPHF DLSH     GS
HAOR       MLTDAEKKEVTALWGKAAGHGEEYGAEALERLFQAFPTTKTYFSHF DLSH     GS
HADK       VLSAADKTNVKGVFSKIGGHAEEYGAETLERMFIAYPQTKTYFPHF DLSH     GS
HBHU      VHLTPEEKSAVTALWGKV   NVDEVGGEALGRLLVVYPWTQRFFESFGDLSTPDAVMGN
HBOR      VHLSGGEKSAVTKLWGKV   NINELGGEALGRLLVVYPWTQRFFEAFGDLSSAGAVMGN
HBDK      VHWTAEEKQLITGLWGKV   NVADCGAEALARLLIVYPWTQRFFASFGNLSSPTAILGN
MYHU       GLSDGEWQLVLNVWGKVEADIPGHGQEVLIRLFKGHPETLEKFDKFKHLKSEDEMKAS
MYOR       GLSDGEWQLVLKVWGKVEGDLPGHGQEVLIRLFKTHPETLEKFDKFKGLKTEDEMKAS
IGLOB     SPLTADEASLVQSSWK    AVSHNEVEILAAVFAAYPDIQNKFSQFA1GKDLASIKDT
GPUGNI     ALTEKQEALLKQSWEVLKQNIPAHSLRLFALIIEAAPESKYVFSFLKDSNEIPE  NN
GPYL      GVLTDVQVALVKSSFEEFNANIPKNTHRFFTLVLEIAPGAKDLFSFLKGSSEVPQ  NN
GGZLB      MLDQQTINIIKATVPVLKEHGVTITTTFYKNLFAKHPEVRPLF    DMGRQE   SL

            vvvvv                                vvvv*

HUMA      AQVKGHGKKVADALTNAV      AHVDDM   PNALSALSDLHAHKLRVDPVNFKLLS
HAOR      AQIKAHGKKVADALSTAA      GHFDDM   DSALSALSDLHAHKLRVDPVNFKLLA
HADK      AQIKAHGKKVAAALVEAV      NHVDDI   AGALSKLSDLHAQKLRVDPVNFKFLG
HBHU      PKVKAHGKKVLGAFSDGL      AHLDNL   KGTFATLSELHCDKLHVDPENFRLLG
HBOR      PKVKAHGAKVLTSFGDAL      KNLDDL   KGTFAKLSELHCDKLHVDPENFNRLG
HBDK      PMVRAHGKKVLTSFGDAV      KNLDNI   KNTFAQLSELHCDKLHVDPENFRLLG
MYHU      EDLKKHGATVLTALGGIL      KKKGHH   EAEIKPLAQSHATKHKIPVKYLEFIS
MYOR      ADLKKHGGTVLTALGNIL      KKKGQH   EAELKPLAQSHATKHKISIKFLEYIS
IGLOB     GAFATHATRIVSFLSEVIAL1SGNTSNAAAV  NSLVSKLGDDHKARGVSAAQ1FGEFR
GPUGNI    PKLKAHAAVIFKTICESA   TELRQKGHAVWDNNTLKRLGSIH LKNKITDPHFEVMK
GPYL      PDLQAHAGKVFKLTYEAA    IQLEVNGAVASDATLKSLGSVHVSKGVVDA HFPVVK
GGZLB     EQPKALAMTVLAAAQNI       ENLPAI   LPAVKKIAVKHC QAGVAAAHYPIVG

            vvvvv             vvv

HUMA      HCLLVTLAAHLPAEFTPAVHASLDKFLASVSTVLTSKYR
HAOR      HCILVVLARHCPGEFTPSAHAAMDKFLSKVATVLTSKYR
HADK      HCFLVVVAIHHPAALTPEVHASLDKFMCAVGAVLTAKYR
HBHU      NVLVCVLAHHFGKEFTPPVQAAYQKVVAGVANALAHKYH
HBOR      NVLIVVLARHFSKDFSPEVQAAWQKLVSGVAHALGHKYH
HBDK      DILIIVLAAHFTKDFTPECQAAWQKLVRVVAHALARKYH
MYHU      ECIIQVLQSKHPGDFGADAQGAMNKALELFRKDMASNYKELGFQG
MYOR      EAIIHVLQSKHSADFGADAQAAMGKALELFRNDMAAKYKEFGFQG
IGLOB     TALVAYLQANVS  WGDNVAAAWNKAL1DNTFAIVVPRL
GPUGNI    GALLGTIKEAIKENWSDEMGQAWTEAYNQLVATIKAEMKE
GPYL      EAILKTIKEVVGDKWSEELNTAWTIAYDELAIIIKKEMKDAA
GGZLB     QELLGAIKEVLGDAATDDILDAWGKAYGVIADVFIQVEADLYAQAVE
```

Figure 4.1
A multiple alignment of several amino acid sequences of globin proteins modified from the paper of McLure, Vasi, and Fitch (316).

The second example is concerned with the cystic fibrosis (CF) gene (Waterman 1995). Cystic fibrosis is an autosomal recessive genetic disorder affecting a number of organs (Riordan et al. 1989). Two long repeated regions of the CF gene sequence (known as CFTR), R_N, starting at position nearer the N terminus, and R_C, starting at position nearer the C terminus, have been identified using a computer program. Via a database search, a number of similar sequences have been found. The names of these sequences are omitted here. The search has highlighted similarities of CFTR to a family of related ATP binding proteins that were already discovered and studied. In figure 4.2, we illustrate a multiple alignment of the repeated CFTR regions and some selected ATP binding sequences that are of high similarity to CFTR. These selected sequences align very well to R_N or R_C. This tells us that R_N and R_C comprise two ATP binding sites in CFTR. If CFTR had been found similar to only one of the members of the ATP binding family, or if the similarity had not been to the ATP binding sites, then these powerful conclusions could not have been so easily drawn.

In this chapter, we discuss some of the most popular mathematical models for multiple sequence alignment and efficient algorithms for computing optimal multiple alignment under these models. Due to the space constraint, we will focus on recent advances in combinatorial (as opposed to stochastic) algorithms, and leave many other important results on multiple sequence alignment untouched. Some surveys and reviews on multiple sequence alignment can be found in Apostolico and Giancarlo 1998; Chan et al. 1992; and McClure et al. 1994.

Section 4.2 presents some basic definitions and several popular mathematical (in fact, combinatorial optimization) models for multiple sequence alignment. Section 4.3 gives some hardness results, demonstrating that computing optimal multiple alignments under these models is computationally difficult. Section 4.4 discusses exact algorithms that give optimal solutions. We then present some approximation algorithms with guaranteed performance in section 4.5 and heuristic algorithms that are popular in practice in section 4.6—in particular, the algorithms in programs Clustal W and GCG and the Gibbs sampling technique. Some concluding remarks and open problems are given in section 4.7.

4.2 Optimization Models for Multiple Sequence Alignment

Given a set of k, $k \geq 2$, sequences, a *multiple alignment* $\mathcal{A}$ is obtained as follows: spaces are inserted into each sequence so that the resulting sequences s_i' ($i = 1, 2, \ldots, k$) have the same length m, and the sequences are arranged in k rows of l columns each. Each column of the alignment contains a space or a letter (nucleotide or

```
hlyb_provu        ISELREGYNTIVGEQGAGLSGGQRQRIAIARALVNNPK-ILIFDEATSALDYESEHVIMRN-MHKICQGRTVIIIAHRLSTVKNADRIIVME
lktb_pasha        ISELREGYNTIVGEQGAGLSGGQRQRIAIARALVNNPK-ILIFDEATSALDYESEHIIMQN-MQKICQGRTVILIAHRLSTVKNADRIIVME
lktb_actac        ISELREGYNTVVGEQGAGLSGGQRQRIAIARALVNNPR-ILIFDEATSALDYESENIIMHN-MHKICQNRTVLIIAHRLSTVKNADRIIVMD
mdr3_human  a     IMKLPQKFDTLVGERGAQLSGGQKQRIAIARALVRNPK-ILLLDEATSALDTESEAEV-QAALDKAREGRTTIVIAHRLSTVRNADVI
hly2_ecoli        ISELREGYNTIVGEQGAGLSGGQRQRIAIARALVNNPK-ILIFDEATSALDYESEHVIMRN-MHKICKGRTVIIIAHRLSTVKNADRIIVME
hlyb_ecoli        ISELREGYNTIVGEQGAGLSGGQRQRIAIARALVNNPK-ILIFDEATSALDYESEHVIMRN-MHKICKGRTVIIIAHRLSTVKNADRIIVME
mdr1_crigr  b     IESLPDKYNTRVGDKGTQLSGGQKQRIAIARALVRQPH-ILLLDEATSALDTESEKVVQEA-LDKAREGRTCIVIAHRLSTIQNADLIVVIQN
mdr2_crigr  b     IESLPDKYNTRVGDKGTQLSGGQKQRIAIARALVRQPH-ILLLDEATSALDTESEKVVQEA-LDKAREGRTCIVIAHRLSTIQNADLIVVIQN
mdr1_human  b     IESLPNKYSTKVGDKGTQLSGGQKQRIAIARALVRQPH-ILLLDEATSALDTESEKVVQEA-LDKAREGRTCIVIAHRLSTIQNADLIVVFQN
mdr1_mouse  b     IDSLPDKYNTRVGDKGTQLSGGQKQRIAIARALVRQPH-ILLLDEATSALDTESEKVVQEA-LDKAREGRTCIVIAHRLSTIQNADLIVVIEN
mdr3_crigr        IMKLPQKFDTLVGERGAQLSGGQKQRIAIARALVRNPK-ILLLDEATSALDTESEAEV-QAALDKAREGRTTIVIAHRLSTVRNADVI
msba_ecoli        INKMDNGLDTVIGENGVLLSGGQRQRIAIARALLRDSP-ILILDEATSALDTESERAI-QAALDELQKNRTSLVIAHRLSTIEKADEIVVVED
ste6_yeast  a               ETLIGTGGVTLSGGQQQRVAIARAFIRDTP-ILFLDEAVSALDIV-HRNLLMKAIRHWRKGKTTIILTHELSQIESDDYLYLMKE
mdr2_mouse  a     IMKLPQKFDTLVGDRGAQLSGGQKQRIAIARALVRNPK-ILLLDEATSALDTESEAEV-QAALDKAREGRTTIVIAHRLSTIRNADVI
heta_anasp                    LGDRGVRLSGGQRQRIAIARALLRDPE-ILILDEATSALDSVSERLIQES-IEKLSVGRTVIAIAHRLSTIAKADKVVVMEQ
mdr_leita   a LEADLAQFCGGLDTEIGEMGVNLSGGQKARVSLARAVYANRD-VYLLDDPLSALDAHVGQRIVQDVILGRLRGKTRVLATHQIHLLPLADYIVVL
cftr_squac  a LEEDITVFPNKDKTVLGDGGITLSGGQRARISLARALYKDAD-LYLLDSPFSHLDVTTEKDIFESCLCKLMVNKTRILVTSKLEHLKKADKILLLHE
cftr_xenla  a LEEDISKFPEKDNTVLGEGGITLSGGQRARISLARAVYKDAD-LYLLDSPFSYLDLFTEKEIFESCVCKLMANKTRILVTSKVEQLKKADKVLILHE
cftr_mouse  a LQQDITKFAEQDNTVLGEGGVTLSGGQRARISLARAVYKDAD-LYLLDSPFGYLDVFTEEQVFESCVCKLMANKTRILVTSKMEHLRKADKILILHQ
cftr_human  a LEEDISKFAEKDNIVLGEGGITLSGGQRARISLARAVYKDAD-LYLLDSPFGYLDVLTEKEIFESCVCKLMANKTRILVTSKMEHLKKADKILILNE
                  |  |  |   ||  ||  ||        ||| |   |    ||| |   ||  |   |         |  | ||     |       |   |
cftr_human  b LRSVIEQFPGKLDFVLVDGGCVLSHGHKQLMCLARSVLSKAK-ILLLDEPSAHLDPVTY-QIIRRTLKQAFADCTVILCEHRIEAMLECQQFLVIEE
cftr_mouse  b LKSVIEQFPGQLNFTLVDGGYVLSHGHKQLMCLARSVLSKAK-IILLDEPSAHLDPITY-QVIRRVLKQAFAGCTVILCEHRIEAMLDCQRFLVIEE
cftr_xenla  b LKLIIDQFPGQLDFVLLDGGCVLSHGHKQLVCLARSVLSKAK-ILLLDEPSAHLDPITF-QIIRKTLKHAFADCTVILSEHRLEAMLECQRFLVIED
cftr_squac  b LKSMIEQFPDKLNFVLVDGGYILSNGHKQLMCLARSILSKAK-ILLLDEPTAHLDPVTF-QIIRKTLKHTFSNCTVILSEHRVEALLECQQFLVIE
mdr_leita   b                 VLEGGSNYSVGQRQLMCMARALLKRGSGFILMDEATANIDPALD-RQIQATVMSAFSAYTVITIAHRLHTVAQYDKIIVMD
mdr2_mouse  b     IETLPQKYNTRVGDKGTQLSGGQKQRIAIARALIRQPR-VLLLDEATSALDTESE-KVVQEALDKAREGRTCIVIAHRLSTIQNADLIVVIEN
ste6_yeast  b                          LLSGGQAQRLCIARALLRKSK-ILILDECTSALDSVSS-SIINEIVKKGPPALLTMVITHSEQMMRSCNSIAVLKD
cyab_borpe          QLPEGYDTMLGENGVGLSGGQRQRIGIARALIHRPR-VLILDEATSALDYESE-HIIQRNMRDICDGRTVIIIAHRLSAV-RCADRIVVME
mdr1_human  a     IMKLPHKFDTLVGERGAQLSGGQKQRIAIARALVRNPK-ILLLDEATSALDTESE-AVVQVALDKARKGRTTIVIAHRLSTV
mdr3_human  b     IETLPHKYETRVGDKGTQLSGGQKQRIAIARALIRQPQ-ILLLDEATSALDTESE-KVVQEALDKAREGRTCIVIAHRLSTIQNADLIVVFQN
hlyb_actpl        ISELREGYNTIVGEQGAGLSGGQPNRIAIARALVNNPK-ILIFDEATSALDYESE-HIIMRNMHQICKGRTVIIIAHRLSTVKNAASIIVME
```

Figure 4.2
Local alignment to R_N and R_C.

The given sequences are: s_1:ACTG s_2:ATCG s_3:GCCA s_4:GTTA

A possible multiple alignment:

s_1'	A	C	T		G
s_2'	A		T	C	G
s_3'	G	C	C		A
s_4'	G	T	T		A

Figure 4.3
An example multiple alignment of four DNA sequences.

amino acid) from each sequence. Figure 4.3 shows an example of multiple sequence alignment.

The *cost* of the multiple alignment $\mathcal{A}$ formed above is defined as

$$\sum_{i=1}^{l} \mu(s_1'(i), s_2'(i), \ldots, s_k'(i))$$

where $s_j'(i)$ denotes the i-th letter in the resulting sequence s_j', $j = 1, 2, \ldots, k$, and $\mu(s_1'(i), s_2'(i), \ldots, s_k'(i))$ denotes the cost of the i-th column. In general, the cost of a column reflects the degree of *dissimilarity* among letters in the column. The multiple sequence alignment problem is to construct a multiple alignment *minimizing* its cost.[1]

The multiple sequence alignment problem is combinatorial in nature because there are exponentially many ways of inserting spaces to form an alignment, even if we limit the length of the alignment. Clearly, the cost of a multiple alignment $\mathcal{A}$ is uniquely determined by the column cost function $\mu(\ )$. Many forms of column cost have been proposed in the literature, resulting in different models for multiple sequence alignment as a combinatorial optimization problem. In the following, we only introduce the most popular models (Altschul and Lipman 1989; Carrillo and Lipman 1988; Gusfield 1993, 1997; Sankoff and Kruskal 1983; Waterman 1995): SP alignment, consensus alignment (also called *star alignment*), and tree alignment. In all of these models, the column cost function $\mu(\ )$ is defined in terms of costs between pairs

1. It is popular among biologists to consider the *score* of a column instead that reflects the degree of similarity among letters in the column, and attempt to find a multiple alignment to *maximize* its total score. The two forms of multiple alignment are easily seen as equivalent in terms of the optimal solution (actually, complementary) optimization problems (Waterman 1989). For simplicity, we will discuss all results in terms of cost and minimization.

of letters and spaces, and we assume that we are given a pairwise cost function (also called *cost scheme*), also denoted as $\mu(a, b)$, that measures the dissimilarity between a pair of letters or spaces a and b. More discussion on popular pairwise cost schemes will be given later in this section.

Throughout this paper, we use Δ to denote a space and Σ to denote the set (i.e., alphabet) of letters that form input sequences.

4.2.1 SP Alignment

In this model, the cost of the i-th column of alignment $\mathcal{A}$ is defined as:

$$\mu(s'_1(i), s'_2(i), \ldots, s'_k(i)) = \sum_{1 \le p < q \le k} \mu(s'_p(i), s'_q(i))$$

where $\mu(s'_p(i), s'_q(i))$ is the cost of the two opposing letters $s'_p(i)$ and $s'_q(i)$ in the column. This column cost function is often referred to as the *Sum-of-all-Pairs* (or SP) cost.

Example 1 Assume that in the pairwise cost scheme $\mu(\ )$, a match costs 0 and a mismatch costs 1: $\mu(a, b) = 0$ if $a = b$ and $\mu(a, b) = 1$ otherwise. The SP-cost of the first column of the alignment given in figure 4.3 is $2 + 2 + 0 = 4$. The SP-costs of the second, third, fourth, and fifth columns are $2 + 2 + 1 = 5$, $1 + 1 + 1 = 3$, $1 + 2 + 0 = 3$, and $2 + 2 + 0 = 4$, respectively. Therefore, the total SP-cost of the alignment is $4 + 5 + 3 + 3 + 4 = 19$.

SP alignment is a useful model in applications such as finding conserved regions, and has previously been studied extensively (Baconn and Anderson 1986; Bafna et al. 1997; Carrillo and Lipman 1988; Gupta et al. 1995; Gusfield 1993; Li et al. 2000; Lipman et al. 1989; Pevzner 1992; Schuler et al. 1991). In an SP alignment, each sequence is assumed to be equally related to all other sequences and thus all pairs of sequences are given the same weight in the definition of alignment cost. This makes sense when all sequences are closely related to each other or when the relationship between the sequences are not known. On the other hand, SP alignment would not be an appropriate model when the sequences considered contain both closely related and remotely related sequences, such as in applications such as phylogeny reconstruction.

4.2.2 Consensus Alignment

The *consensus* cost of the i-th column of alignment $\mathcal{A}$ is defined as follows:

$$\mu(s'_1(i), s'_2(i), \ldots, s'_k(i)) = \min_{s \in \Sigma \cup \{\Delta\}} \sum_{j=1}^{n} \mu(s'_j(i), s)$$

The given sequences are : s_1: ACTG s_2:ATCG s_3:GCCA s_4:GTTA

A possible multiple alignment:

	s'_1	A	C	T		G
	s'_2	A		T	C	G
	s'_3	G	C	C		A
	s'_4	G	T	T		A
Consensus sequence		A	C	T		G
Another consensus sequence		G	C	T		A

Figure 4.4
An example alignment and its consquence sequences.

In other words, to define the cost of $\mathscr{A}$, we attempt to reconstruct a letter or space for each column of $\mathscr{A}$ and thus obtain a *consensus* sequence that has the smallest over cost to all input sequences. The consensus sequence can be used as a representative of the input sequences in various applications (Gusfield 1997).

Example 2 Again, assume that a match costs 0 and a mismatch costs 1 in the pair-wise cost scheme. Consider the same four sequences as in example 1. Two consensus sequences for $\mathscr{A}$ are shown in figure 4.4. (Note that the consensus sequence may not be unique for an alignment.) The cost of the first column is 2. The costs of the second, third, fourth and fifth columns are 2, 1, 1, and 2, respectively. Hence, the total consensus cost of $\mathscr{A}$ is $2 + 2 + 1 + 1 + 2 = 8$.

4.2.3 Tree Alignment

In order to define the *tree-cost* of alignment $\mathscr{A}$, an *evolutionary* (or *phylogenetic*) tree $T = (V, E)$ with k leaves is assumed to be given, where V and E denote the sets of nodes and edges of T. Each leaf j of T corresponds to an input sequence s_j. Let $k+1$, $k+2, \ldots, k+m$ be the internal nodes of T. To obtain the tree-cost $\mu(s'_1(i), s'_2(i), \ldots, s'_k(i))$ of the i-th column of $\mathscr{A}$, we need reconstruct a letter or space $s'_j(i)$ for each internal node j, such that

$$\sum_{(p,q)\in E} \mu(s'_p(i), s'_q(i))$$

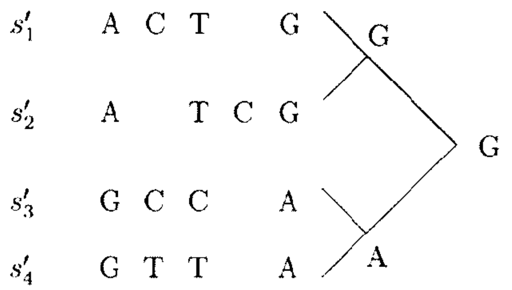


Figure 4.5
An optimal assignment of nucleotides at internal nodes for the rightmost column.

is minimized. The tree-cost of the column is thus defined as

$$\mu(s_1'(i), s_2'(i), \ldots, s_k'(i)) = \sum_{(p,q) \in E} \mu(s_p'(i), s_q'(i))$$

In other words, if we think of the pairwise cost scheme $\mu(a, b)$ as a measure of the "evolutionary cost" from a to b, then the tree-cost of the i-th column of alignment $\mathcal{A}$ is really the least cost to generate the letters or spaces $s_1'(i), s_2'(i), \ldots, s_k'(i)$ via the given evolutionary tree T.

Although computing the tree-cost of a column of $\mathcal{A}$ is nontrivial because it involves assigning a letter or space to each internal node of T optimally, there is a simple dynamic programming algorithm that computes tree-cost for a column of k letters/spaces in $O(k|\Sigma|)$ time (Sankoff and Rousseau 1975). (See section 4.4.1 for more details.) Figure 4.5 illustrates a multiple alignment of four DNA sequences, a given evolutionary tree connecting the sequences, and an optimal assignment of letters (in this case, nucleotides) at the internal nodes of the tree for the rightmost column of the alignment, assuming the same simple pairwise cost scheme as in examples 1 and 2. The tree-cost of the column is thus 1.

This model of multiple alignment was first studied by D. Sankoff in 1975 (Sankoff 1975). Since then, multiple sequence alignment with tree-cost has received a lot of attention in the computational molecular biology community (Altschul and Lipman 1989; Hein 1989; Jiang et al. 1994; Sankoff 1975; Sankoff et al. 1976; Sankoff and Kruskal 1983; Wang et al. 1996). For simplicity, multiple sequence alignment with tree-cost is often referred to as *tree alignment* in the literature (Sankoff 1975).

It is easy to see that if we consider arbitrary evolutionary trees T in tree alignment, then consensus alignment is in fact a special case of tree alignment where the evolutionary tree is simply a *star* with one internal node connecting to all leaves. However, in practical evolutionary trees, internal nodes often have bounded degrees. In fact, most evolutionary trees studied in the literature are binary trees.

Note that a tree alignment induces a set of *reconstructed* sequences, one for each internal node of the tree T. Thus, it is convenient to reformulate tree alignment as a sequence reconstruction problem as follows.

4.2.4 An Alternative Formulation of Tree Alignment

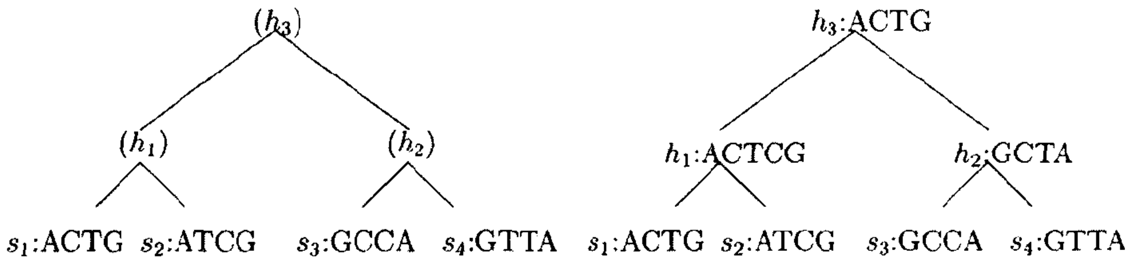
Suppose that we are given k sequences and a (rooted) evolutionary tree containing k leaves, each of which is labeled with a unique given sequence. The problem is to assign a sequence to each internal node of the tree such that the *cost* of the resulting *fully labeled* tree is minimized. Here, the cost of a fully labeled tree is the total cost of its edges and the cost of an edge is the optimal pairwise alignment cost between the two sequences associated with both ends of the edge.

Observe that once a sequence for each internal node has been reconstructed, a multiple alignment can be obtained by optimally aligning the pair of sequences associated with each edge of the tree. Moreover, the tree-cost of this *induced* multiple alignment equals the cost of T as defined above. In this sense, the two formulations of tree alignment are equivalent.

We demonstrate this by an example in which the sequences inferred for the internal nodes of an evolutionary tree actually induce a multiple alignment of the k leaf sequences. Such a multiple alignment is likely to expose significant evolutionary relationships among the leaf sequences, according to the maximum parsimony principle.

Example 3 Consider again four given sequences ACTG, ATCG, GCCA, and GTTA, and the evolutionary tree shown in figure 4.6a connecting these sequences. Suppose that the internal sequences are reconstructed as in figure 4.6b. For each edge of the tree in (b) we can construct an optimal pairwise alignment of the two sequences associated with the edge. Then we can induce a multiple alignment that is *consistent* with all six pairwise alignments by "merging" the pairwise alignments incrementally (Feng and Doolittle 1987; Thompson et al. 1994), taking the spaces embedded in each sequence into consideration. For example, from the given pairwise alignments of sequences s_1, h_1 and of sequences s_2, h_1, we can obtain a multiple alignment of s_1, s_2, h_1 as shown in figure 4.6d. The final induced multiple alignment of the four leaf sequences is shown in figure 4.6e (if we ignore everything below the line).

The biological interpretation of the model is that the given tree represents the evolutionary history (known by means other than sequence analysis or postulated in a phylogeny reconstruction procedure) that has created the biomolecular (DNA, RNA, or amino acid) sequences written at the leaves of the tree. The leaf sequences are ones found in organisms existing today and the sequences to be determined at the internal nodes of the tree represent inferred sequences that may have existed in the ancestral organisms.



(a) The input sequences and tree. (b) The fully labeled tree.

s'_1:	A	C	T		G
h_1:	A	C	T	C	G

s'_2:	A		T	C	G
h_1:	A	C	T	C	G

s'_3:	G	G	C	A
h_2:	G	C	T	A

s'_4:	G	T	T	A
h_2:	G	C	T	A

h_1:	A	C	T	C	G
h_3:	A	C	T		G

h_2:	G	C	T	A
h_3:	A	C	T	G

(c) The pairwise alignments.

s'_1:	A	C	T		G
h_1:	A	C	T	C	G
s'_2:	A		T	C	G

(d) The induced alignment of s_1, s_2 and h_1.

s_1	A	C	T		G
s_2	A		T	C	G
s_3	G	C	C		A
s_4	G	T	T		A
h_1	A	C	T	C	G
h_2	G	C	T		A
h_3	A	C	T		G

(e) The induced multiple alignment

Figure 4.6
Reconstructed Sequences at the internal nodes induce a multiple alignment.

We note in passing that some variants of tree alignment have also been studied in the literature. Tree alignment with *recombination* has recently been proposed by Wang et al. (2000). It takes recombination events into consideration, in addition to the usual substitution, insertion, and deletion events. Another variant, *generalized tree alignment*, was studied by Hein (1989, 1990), Sankoff and Kruskal (1983), Schwikowski and Vingron (1997), and Wang and Jiang (1994). In generalized tree alignment, we are only given k sequences and we are supposed to construct an evolutionary tree T as well as a multiple sequence alignment minimizing its tree-cost (or, equivalently, a sequence at each internal node of T minimizing the cost of the resulting fully labeled tree).

We end the section with a brief discussion on pairwise cost schemes and some key concepts in approximation algorithms.

4.2.5 Pairwise Cost Schemes

The choice of cost schemes for pairs of letters is an important issue in sequence analysis. In general, such a cost scheme reflects the probabilities of evolutionary events, including substitution, insertion, and deletion. For protein sequences, the PAM matrix and BLOSUM matrix are the most popular ones (Henikoff and Henikoff 1992; Schwarz and Dayhoff 1979).[2] For DNA sequences, the simple match/mismatch cost scheme mentioned in examples 1–3 is often used. More sophisticated cost schemes include transition/transversion costs (Sankoff et al. 1976) and DNA PAM matrices.

From a computational point of view, cost schemes that satisfy the following conditions are especially interesting:

(C1) $\mu(a,a) = 0$

(C2) $\mu(a,b) = \mu(b,a)$

(C3) $\mu(a,c) \le \mu(a,b) + \mu(b,c)$ for any c

Such a cost scheme is called a *metric cost* scheme (Sankoff and Kruskal 1983). Metric cost schemes are popular because they enable us to design efficient approximation algorithms with guaranteed performance.

4.2.6 Basics of Approximation Algorithms

We assume that the readers are familiar with the NP-hardness theory (see Garey and Johnson 1979 for a complete treatment). If a problem is NP-hard, then it is unlikely

2. Again, these matrices in fact contain similarity scores rather than dissimilarity costs. The scores can be easily converted into costs so that optimal alignments are the same under both objective functions (i.e., maximizing the score or minimizing the cost).

for us to have a polynomial time algorithm to exactly solve the problem. For some optimization problems, we might be interested in designing *approximation algorithms* that produce solutions with a cost close to that of the optimum.

Let A be an approximation algorithm for a minimization problem. (The definition for maximization problems is symmetric.) The *performance ratio* of A is defined as a number r such that for any instance I of the problem,

$$\frac{A(I)}{OPT(I)} \le r \tag{4.1}$$

where $A(I)$ is the cost of the solution—for instance, I produced by algorithm A—and $OPT(I)$ is the cost of an optimal solution for instance I. An *approximation scheme* for a minimization problem is an algorithm A that takes as input both instance I and an error bound ϵ, and achieves the performance ratio

$$R_A(I, \epsilon) = \frac{A(I)}{OPT(I)} \le 1 + \epsilon$$

Such an algorithm A can in fact be viewed as a family of algorithms $\{A_\epsilon \mid \epsilon > 0\}$, for each error bound ϵ. A *polynomial time approximation scheme* (PTAS) is an approximation scheme $\{A_\epsilon\}$, where the algorithm A runs in time polynomial in the size of the instance I, for any fixed ϵ. (For more details on approximation algorithms and schemes, see Garey and Johnson 1979.)

In terms of approximability of problems, the best one can hope for are PTASs. Some problems have good approximation algorithms, such as PTASs, whereas some other problems are hard to approximate. If a problem is *MAX SNP-hard*, then it is unlikely to have a PTAS. Recently, the theory on inapproximability has been developed (see Hochbaum 1996).

4.3 Hardness Results

In this section, we summarize the computational complexity of computing optimal multiple sequence alignment under the models described in the previous section. These results give motivation for studying approximation algorithms for these problems in section 4.5.

SP alignment was proved to be NP-hard (Jiang et al. 1994; Wang and Jiang 1994). Thus, it is unlikely to be solved in polynomial time. However, the pairwise cost scheme used in this proof does not satisfy the triangle inequality, and is thus not a metric. P. Bionizzoni and G. Della Vedova recently strengthened the result and proved that SP

alignment is NP-hard even if the alphabet size is 2 and the pairwise cost scheme is a metric.

THEOREM 1 (Bonizzoni and Della Vedova 2000) SP Alignment is NP-hard for the case where the alphabet size is 2 and the cost scheme is metric.

The pairwise cost scheme (Bonizzoni and Vedova to appear) is a metric cost scheme using numbers 0, 1, and 2, and the alphabet size is 2. The proof is quite involved.

W. Just has recently proved that SP alignment is NP-hard for the case where insertions of spaces are restricted to both ends of the sequences (Just 1998). The pairwise cost scheme used in the proof is also a metric. A slightly stronger result is given in Li et al. 2000.

THEOREM 2 (Li et al. 2000) SP Alignment is NP-hard when all insertions of spaces are restricted to both ends of the sequences and the pairwise cost scheme has the simplest form, i.e., a match costs 0 and a mismatch costs 1.

Tree alignment was shown to be NP-hard (Jiang et al. 1994). The cost scheme used in the proof is a metric cost scheme using numbers 0, 1, and 2 and the alphabet size is 4.

THEOREM 3 Tree alignment is NP-hard even when the given phylogeny is a binary tree.

For consensus alignment, we have the following theorems:

THEOREM 4 (Li et al. 1999) Consensus alignment is NP-hard when the alphabet size is 4 and the cost scheme has the simplest form, i.e., a match costs 0 and a mismatch costs 1.

Consensus alignment is also hard in terms of approximation.

THEOREM 5 (Jiang et al. 1994; Wang and Jiang 1994) Consensus alignment is MAX SNP-hard if the pairwise cost scheme is arbitrary.

This means that it is unlikely to have a PTAS for consensus alignment if the cost scheme is arbitrary. The cost scheme used in the proof of the theorem does not satisfy the conditions (C1) and (C3) of a metric cost.

4.4 Exact Algorithms

The hardness results in the previous section imply that exact algorithms for the models of multiple alignment described in section 4.2 have to run in exponential time.

However, these exact algorithms are sometimes useful when the number of sequences involved is not too large and the sequences have moderate lengths.

4.4.1 Dynamic Programming in k Dimensions

An optimal solution for a multiple sequence alignment of k given sequences can be obtained by a standard dynamic programming algorithm. Gusfield (1997) and Sankoff and Kruskal (1983) offer extensive discussion of such algorithms.

Let $s_1, s_2, \ldots, s_k$ be k given sequences, each of length m (for simplicity). Let $s_i[j, l]$ denote the substring of s_i containing the letters j through l, and $s_i[j]$ denote the j-th letter of s_i. If $j > l$, $s_i[j, l]$ is empty. Let $d(i_1, i_2, \ldots, i_k)$ be the cost of an optimal alignment for the k prefixes $s_1[1, i_1], s_2[1, i_2], \ldots, s_k[1, i_k]$. All the values $d(i_1, i_2, \ldots, i_k)$ together form a k-dimensional matrix containing $(m+1)^k$ cells, where each i_j may take on $m+1$ values $0, 1, \ldots, m$. We can use the following recurrence equation to compute $d(i_1, i_2, \ldots, i_k)$.

$$d(i_1, i_2, \ldots, i_k) = \min\{d(i'_1, i'_2, \ldots, i'_k) + \mu(s_1[i'_1 + 1, i_1], s_2[i'_2 + 1, i_2], \ldots, s_k[i'_k + 1, i_k])\} \tag{4.2}$$

where each i'_j is either i_j or $i_j - 1$, the operator min is taken among the $2^k - 1$ possible configurations (the configuration where all i'_j are i_j is excluded), $\mu(s_1[i'_1 + 1, i_1], s_2[i'_2 + 1, i_2], \ldots, s_k[i'_k + 1, i_k])$ is the cost of the last column in the alignment for the k prefixes $s_1[1, i_1], s_2[1, i_2], \ldots, s_k[1, i_k]$, containing k letters/spaces, one from each given sequence.

A cell $d(i_1, i_2, \ldots, i_k)$ of the k dimensional matrix is called a *boundary* cell if at least one of its index i_j is 0. From formula (4.2), we know that computing $d(i_1, i_2, \ldots, i_k)$ needs the values of its $2^k - 1$ neighbors in the matrix. Thus, if the value $\mu(s_1[i'_1 + 1, i_1], s_2[i'_2 + 1, i_2], \ldots, s_k[i'_k + 1, i_k])$ and the values of the boundary cells are known, we can compute the values of all the cells in the matrix one by one in the order as suggested by the above recurrence relation. Similar to the pairwise alignment (Gusfield 1997), a standard back-tracing process gives the actual multiple alignment.

THEOREM 6 If $\mu(s_1[i'_1 + 1, i_1], s_2[i'_2 + 1, i_2], \ldots, s_k[i'_k + 1, i_k])$ is known, then the above dynamic programming algorithm runs in $O(2^k m^k)$ time for all the three models multiple alignment: SP alignment, consensus alignment, and tree alignment.

Now, we will explain how to compute the value $\mu(s_1[i'_1 + 1, i_1], s_2[i'_2 + 1, i_2], \ldots, s_k[i'_k + 1, i_k])$ and the values of boundary cells.

SP Score For SP alignment, the values of boundary cells are computed recursively. If $i_1 = 0$, we can use the following equation:

$$d(0, i_2, \ldots, i_k) = d(i_2, \ldots, i_k) + \sum_{i=2}^{k} \sum_{j=1}^{i_j} \mu(s_i[j], \Delta) \tag{4.3}$$

where $\mu(s_i[j], \Delta)$ is the cost of deleting letter $s_i[j]$. The case where $i_j = 0$, $j > 1$, is similar.

Let $A_1, A_2, \ldots, A_k$ be k letters or spaces. The computation of $\mu(A_1, A_2, \ldots, A_k)$ is trivial:

$$\mu(A_1, A_2, \ldots, A_k) = \sum_{1 \le j < l \le n} \mu(A_j, A_l) \tag{4.4}$$

where $\mu(A_j, A_l)$ is the cost for the two opposing letters or spaces A_j and A_l.

Tree Score For tree alignment, the value for boundary cell is computed recursively. For example, if $i_1 = 0$, we can use the following equation:

$$d(0, i_2, \ldots, i_k) = D(i_2, \ldots, i_k) \tag{4.5}$$

where $D(i_2, \ldots, i_k)$ is the cost of an optimal tree alignment for the k prefixes $s_1[1..0]$, $s_2[1..i_2], \ldots, s_k[1, i_k]$. Here $s_1[1, 0]$ denotes an empty sequence. Note that the number of indices in $D(i_2, \ldots, i_k)$ is now reduced to $k - 1$, and $D(i_2, \ldots, i_k)$ can then be computed recursively in a similar fashion.

Let $A_1, A_2, \ldots, A_k$ be k letters or spaces assigned to the k leaves of the given evolutionary tree T. The computation of $\mu(A_1, A_2, \ldots, A_n)$ also requires a dynamic programming algorithm. Let $\mu(i, B)$ denote the cost of the subtree rooted at node i such that node i is assigned the letter/space B. $\mu(i, B)$ can be computed as follows using the following recurrence relation:

$$\begin{aligned} \mu(i, B) = \min_{B_j \in \Sigma} \{ & \mu(i_1, B_1) + \mu(i_2, B_2) + \cdots + \mu(i_d, B_d) \\ & + \mu(B, B_1) + \mu(B, B_2) + \cdots + \mu(B, B_d) \} \end{aligned} \tag{4.6}$$

where $i_1, \ldots, i_d$ denote the children of node i. Figure 4.7 illustrates ideas involved in the equation (4.6). If i is a leaf node and B is the letter initially assigned to i, then $\mu(i, B) = 0$. Otherwise, $\mu(i, B)$ is set to be infinity. $\mu(i, B)$ can be computed bottom up using dynamic programming in time $O(|T| \cdot |\Sigma|) = O(k|\Sigma|)$.

Consensus alignment can be treated as special case of tree alignment in this context.

4.4.2 Reducing the Computation Volume

Carrillo and Lipman proposed a method to cut down the computational volume of the dynamic programming algorithm for SP alignment (Carrillo and Lipman 1988).

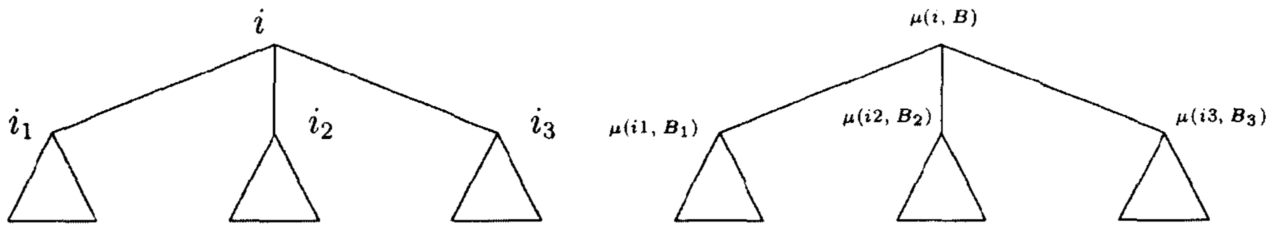


Figure 4.7
(a) A subtree rooted at i. (b) Assigning letters/spaces to node i and its children.

The basic idea is to compute upper bounds on alignment costs for each pair of sequences in the computation of the k dimensional matrix and eliminate those cells of the matrix that violate the upper bounds.

Given an SP alignment $\mathscr{A}$, let $\mathscr{A}_{i,j}$ be the pairwise alignment between sequences s_i and s_j induced by $\mathscr{A}$ (simply get rid of the other sequences in $\mathscr{A}$ and delete columns containing spaces in both sequences s_i and s_j). Let $c(\mathscr{A})$ denote the cost of $\mathscr{A}$ and $c(\mathscr{A}_{i,j})$ the cost of $\mathscr{A}_{i,j}$.

Let $\mathscr{B}$ be an optimal SP alignment and $C \geq c(\mathscr{B})$ an upper bound on the cost $c(\mathscr{B})$. Let $C_{i,j}$ be the optimal alignment cost between sequences s_i and s_j. Then we have, for any $1 < x < y < k$,

$$C \geq c(\mathscr{B}) = \sum_{i<j} c(\mathscr{B}_{i,j}) \geq c(\mathscr{B}_{x,y}) - C_{x,y} + \sum_{i<j} C_{i,j} \tag{4.7}$$

That is,

$$c(\mathscr{B}_{x,y}) \leq C_{x,y} + \left(C - \sum_{i<j} C_{i,j}\right) \tag{4.8}$$

Because $C_{i,j}$ can be computed quickly, if we know how to choose $C \geq c(\mathscr{B})$, we would have an upper bound $C_{x,y} + (C - \sum_{i<j} C_{i,j})$ on $c(\mathscr{B}_{x,y})$. Therefore, in the computation of the k dimensional matrix, we do not have to go through all the cells in the matrix. Instead, we can rule out those cells that violate inequality (4.8). These cells can be identified easily using dynamic programming on s_x and s_y. This approach allows one to be able to optimally align up to six sequences of practical lengths (Carrillo and Lipman 1988). Improved versions of this technique are reported by Gupta et al. (1995) and Altschul and Lipman (1989).

Altschul and Lipman proposed a similar method for tree alignment (Altschul and Lipman 1989). They designed a nice method to compute the upper bound for tree alignment. Moreover, special care was given for consensus alignment as a special

case of tree alignment. We refer interested readers to Altschul and Lipman (1989) for more details.

We emphasize that the speed of the above algorithm depends very much on the quality of the upper bound C, although the algorithm always produces an optimal solution.

4.5 Approximation Algorithms

Because computing optimal SP/tree/consensus alignment is NP-hard and the running times of exact algorithms for these problems are exponential in the number of given sequences k, there has been a lot of interest in finding efficient approximation algorithms for these problems. In this section, we give a summary of recent approximation results on SP alignment, consensus alignment, and tree alignment.

4.5.1 SP Alignment and Diagonal Band

SP alignment has been extensively studied recently. With great effort, the best known approximation ratio for SP alignment has been improved from $2 - \frac{2}{k}$ to $2 - \frac{l}{k}$ for any constant l, where k is the number of the sequences (Bafna et al. 1997; Gusfield 1993; Pevzner 1992). The $2 - o(1)$ barrier on approximation ratio appears to be formidable. A more recent progress is a PTAS for a special case of the problem: *multiple alignment within a band.* The restriction of aligning sequences within a constant diagonal band is often used in practical situations. Methods under this assumption have been extensively studied as well. Sankoff and Kruskal (1983) discussed the problem under the rubric of "cutting corners." Alignment within a band is used in the final stage of the well-known FASTA program for rapid searching of protein and DNA sequence databases (Pearson and Lipman 1988; Pearson 1990). Pearson showed that alignment within a band gives very good results for many protein superfamilies (Pearson 1991). Other references on the subject include Altschul and Lipman (1989), Chao et al. (1992), Fickett (1984), Galil and Ciancarlo (1989), and Ukkonen (1985). Spouge (1991) gives a survey on this topic.

A formal definition of multiple alignment within a constant band is given below.

c-DIAGONAL SP ALIGNMENT Let $\mathcal{S} = \{s_1, s_2, \ldots, s_k\}$ be a set of k sequences, each of length m (for simplicity), and $\mathcal{M}$ an alignment of the k sequences. Let the length of the alignment $\mathcal{M}$ be M. $\mathcal{M}$ is called a *c-diagonal alignment* if for any $p \leq m$ and $1 < i < j < k$, if the p-th letter of s_i is in column q of $\mathcal{M}$ and the p-th letter of s_j is in column r of $\mathcal{M}$, then $|q - r| \leq c$. In other words, the inserted spaces are "evenly"

distributed among all sequences and the i-th position of a sequence is about c positions at most away from the i-th position of any other sequence.

The Center Star Approach for SP Alignment The first approximation algorithm for SP alignment was given by D. Gusfield (Gusfield 1993). He introduced the *center star* method, which is very simple and efficient. The method begins by selecting a sequence (called the *center sequence*) s_c from the set of k given sequences $\mathcal{S}$ such that $\sum_{i=1}^{k} \mu(s_c, s_i)$ is minimized, where $\mu(s_c, s_i)$ denotes the optimal pairwise alignment cost between s_c and s_i. It then optimally aligns the sequences in $\mathcal{S} - \{s_c\}$ to s_c, yielding $k - 1$ pairwise alignments. These $k - 1$ pairwise alignments can be combined together to form a single multiple alignment for all k sequences in $\mathcal{S}$. If the cost scheme for pairs of letters is a metric, the cost of the multiple alignment produced by the center star algorithm is at most twice the optimal cost (Gusfield 1993, 1997).

THEOREM 7 (Gusfield 1993, 1997) The approximation ratio of the center star algorithm is $2 - \frac{2}{k}$ for metric pairwise cost schemes.

The l-star Approach Bafna, Lawer, and Pevzner extended the center star approach to l-stars that improve upon the approximation ratio of the center star algorithm (Bafna et al. 1996). The basic idea is to (1) select a center sequence s_c; (2) decompose the k given sequences into many groups, each of which contains l sequences including sequence s_c; (3) optimally align the l sequences in each group; and (4) combine the multiple sequence alignments of l sequences into an alignment for k sequences using the common sequence s_c.

The difficulty here is how to partition the k given sequences. Pevzner (1992) solved the case for $l = 3$ using graph matching and obtained an approximation ratio $2 - \frac{3}{k}$. Bafna et al. (1996) use an algorithm designed to compute a good set of groups of size l or $l + 1$ sequences.

THEOREM 8 (Bafna et al. 1996) There is an approximation algorithm with performance ratio $2 - \frac{l}{k}$ that runs in time $O(k^{l+1}(2^k + k \cdot g(l, m))$ time, where $g(l, m)$ is the time required to optimally align l sequences of length m.

Another algorithm that has the same performance ratio but runs in $O(k^3 g(2l + 3, m))$ time was also proposed by Bafna et al. (1996). Although this latter algorithm is truly polynomial in both m and k, its running time might be in fact slower than the former one when l is large (compared with k), as optimally aligning $2l + 3$ sequences is very time consuming.

Algorithm RandomAlign

1. **for** $c \in \{1, 2, \ldots, k\}$ **do**

 repeat $2lg(\frac{k}{\epsilon})$ times

 Choose a random partition G with common sequence s_c

 Compute an alignment based on the partition G

 Choose the best alignment among the $2\log(\frac{k}{\epsilon})$ partitions

 Choose the best obtained alignment for $c \in \{1, 2, \ldots, k\}$

Figure 4.8
A randomized algorithm for partitioning sequences.

The above algorithms all require a good method for partitioning the k sequences into groups. A simple and efficient approach is the randomized algorithm given in figure 4.8.

THEOREM 9 If $l - 1$ divides $k - 1$, then for any $\epsilon > 0$, algorithm RandomAlign runs in time $O\left(k^2 \log\left(\frac{k}{\epsilon}\right) g(l, m)\right)$ and achieves a performance ratio $2 - \frac{l}{k}$ with probability $1 - \epsilon$.

A PTAS for c-diagonal SP Alignment A major open problem in multiple sequence alignment is if SP alignment has a PTAS (Jiang et al. 2000). In this section, we sketch a recent PTAS for a restricted version of SP alignment where spaces are inserted "evenly," that is, c-diagonal SP alignment. Our PTAS works for all metric cost schemes, but for the ease of presentation, here we will consider the simple match/mismatch cost scheme (i.e., a match costs 0 and a mismatch costs 1.) We need some definitions first.

DEFINITION 1 Let $s_1, \ldots, s_k$ be sequences and $\mathcal{A}$ a multiple alignment of the sequences. Suppose that $s'_1, \ldots, s'_k$ are the rows of $\mathcal{A}$ containing the padded sequences. If $s'_i[p]$ is Δ, whereas $s'_j[p]$ is not, then position p represents to an insertion in sequence s_j. On the other hand, if $s_i[p]$ is not Δ, whereas $s_j[p]$ is, then position p represents a deletion in s_j.

In a multiple alignment, an insertion may correspond to many deletions and vice versa. Moreover, for a column of alignment, the number of insertions uniquely determines the number of deletions and vice versa. In the construction of our algorithms, we will use the smaller of the numbers of insertions and deletions when we count the total number of insertions and deletions. For example, in the multiple alignment

```
caaccca
ca cccc
ca cccg
ca ccct
```

Figure 4.9
One insertion corresponds to three deletions.

in figure 4.9, the total number of insertions/deletions (or simply *indel*) should be counted as 1 instead of 3 (or 4).

The following restricted version of SP alignment, called Average c-Indel SP Alignment, would serve as a useful intermediate step toward our PTAS for c-diagonal SP alignment.

DEFINITION 2 The Average c-Indel SP Alignment problem is to find a multiple alignment for a given set $\mathcal{S}$ of sequences with the minimum possible SP cost such that on average, there are at most c indels per sequence.

Let l be the length of alignment $\mathcal{A}$ of sequences $s_1, \ldots, s_k$. Let $\lambda_{j,a}$ be the number of the occurrences of letter a at the j-th position of $\mathcal{A}$. The SP-cost of $\mathcal{A}$ can be re-written as:

$$\mu(\mathcal{A}) = \sum_{j=1}^{l} \sum_{\substack{a \neq b \\ a,b \in \Sigma \cup \{\Delta\}}} \lambda_{j,a} \cdot \lambda_{j,b}$$

Clearly, $\frac{\lambda_{j,a}}{k}$ is the frequency of letter (or space) a in the j-th column of the alignment. We call the $l \times (|\Sigma| + 1)$ matrix formed by $\frac{x_{j,a}}{k}$ the *frequency matrix* of $\mathcal{A}$. The frequency matrix is called a *profile* in the literature.

Our algorithm consists of two major steps: (1) Randomly choose (or try all combinations) r sequences from the k sequences, where r is a constant parameter. By trying all possible "feasible" alignments of the r sequences involving at most c indels per sequence, we can suppose that we know the "correct" alignment $\mathcal{A}^r$ of the r sequences that is induced by $\mathcal{A}$. Then we calculate the frequency matrix of $\mathcal{A}^r$, which is hopefully an approximation of the frequency matrix of $\mathcal{A}$. Align every sequence with the frequency matrix of $\mathcal{A}^r$. This can be done by using a slight modification of the standard dynamic programming algorithm for pairwise sequence alignment. The complete algorithm is given in figure 4.10.

Algorithm AverageSPAlign

Input: integer c, h, r and $\mathcal{S} = \{s_1, \ldots, s_k\}$. Each s_i has length m.

Output: a multiple alignment $\mathcal{A}$.

1. **for** l from m to km **do**

   for any $s_{i_1}, s_{i_2}, \ldots, s_{i_r} \in \mathcal{S}$ **do**

   for any possible alignment $\mathcal{A}'$ of $s_{i_1}, s_{i_2}, \ldots, s_{i_r}$ with length l and at most ch indels on each sequence **do**

   (a) Let $\lambda_{j,a}$ be the number of the occurrences of letter a in the j-th column of the alignment $\mathcal{A}'$ for $j = 1, \ldots, l$ and $a \in \Sigma \cup \{\Delta\}$.

   (b) **for** i from 1 to k **do**

   Use dynamic programming to calculate a *padded* s_i' of s_i by inserting spaces such $\sum_{j=1}^{l} \lambda_{j,s_i'[j]}$ is maximized.

   (c) Let $\mathcal{A} = \{s_1', \ldots, s_k'\}$ be a multiple alignment of $s_1, \ldots, s_k$.

   Calculate $\mu(\mathcal{A})$.

2. Output an alignment $\mathcal{A}$ such that $\mu(\mathcal{A})$ is minimized among all $\mathcal{A}$'s obtained in step 1(c).

Figure 4.10
A PTAS for Average c-Indel SP Alignment.

THEOREM 10 If $h \geq 4$, $r \geq 1$, algorithm AverageSPAlign in figure 4.10 outputs an alignment with SP-cost no more than $1 + \frac{2}{r} + \frac{2}{h}$ times the SP-cost of an optimal average c-indel SP alignment.

The proof of theorem 10 is quite involved. The basic idea is to show that if we randomly choose r sequences from $\mathcal{S}$, construct the frequency matrix from the r chosen sequences, and "align" the sequences in $\mathcal{S}$ with the constructed frequency matrix, we then obtain an alignment with an expected cost at most $1 + \frac{2}{r} + \frac{2}{l}$ times of the opti mum (for the complete proof, see Li et al. 2000).

For c-Diagonal SP Alignment, we introduce a new constant parameter, t, that plays a crucial role in cutting the sequences into segments so that on average each segement contains about ct indels in an optimal c-diagonal alignment. The difficulty is that we do not know exactly where to cut the sequences. The c-diagonal condition allows us to find approximate cutting positions so that each segment has at most c

Algorithm DiagonalSPAlign

Input: integers c, h, r and t, and $\mathcal{S} = \{s_1, \ldots, s_k\}$. Each s_i has length m.

Output: a multiple alignment $\mathcal{A}$ of $\{s_1, \ldots, s_k\}$.

1. **for** i from 1 to m **do**

   Let c_i be the cost of the output of algorithm AverageSPAlign on input:

   ct, h, r, and $\{s_1[1..i], \ldots s_k[1..i]\}$.
2. Let I be the maximum i such that $c_i \leq \rho ctk^2$. Remove the first I letters from each s_i as a segment.
3. Repeat steps 1, 2, and 3 until every sequence becomes empty.
4. Concatenate the alignments of the segments to form a multiple alignment for the original set $\mathcal{S}$.

Figure 4.11
A PTAS for c-Diagonal SP Alignment.

"incorrect" letters (either c extra letters or c missing letters) at each end. Because each segment has on average about $O(ct)$ indels, the error rate for each sequence is about $\frac{1}{t}$. After cutting the sequences, we can use the PTAS for Average c-Indel SP Alignment for each group of segments.

In the algorithm, we dynamically cut the input sequences into small segments such that the SP alignment cost for each group of segments is about ctk^2, for some constant t. This can be easily done by trying to cut all the sequences at the i-th position ($i = 1, 2, \ldots m$) and test if such a i leads to a group of segments whose SP alignment cost is about ctk^2.

The complete algorithm is given in figure 4.11. Let $\rho = 1 + \frac{2}{h} + \frac{2}{r}$.

THEOREM 11 Approximation ratio of algorithm DiagonalSPAlign is

$$\rho\left(1 + \frac{2}{t - 2 - \frac{1}{c}}\right).$$

4.5.2 Consensus Alignment

It turns out that the center star approach can also give a ratio 2 approximation algorithm for consensus alignment.

Algorithm AverageConsensusAlign

Input: integers c, h, r, and $\mathcal{S} = \{s_1, \ldots, s_k\}$. Each s_i has length m.

Output: a multiple alignment $\mathcal{A}$.

1. **for** l from m to km **do**

 for any $s_{i_1}, s_{i_2}, \ldots, s_{i_r} \in \mathcal{S}$ **do**

 for any possible alignment $\mathcal{A}'$ of $s_{i_1}, s_{i_2}, \ldots, s_{i_r}$ with length l and at most ch indels on each sequence **do**

 (a) Let S be an optimal consensus sequence implied by $\mathcal{A}'$.

 (b) **for** i from 1 to k **do**

 Using dynamic programming to calculate a padded version s_i' of s_i by inserting spaces such that $|s_i'| = l$ and $d_H(S, s_i')$ (*i.e.*, the Hamming distance) is minimized.

 (c) Let $\mathcal{A} = \{s_1', s_2', \ldots, s_k'\}$. Calculate the consensus cost of this alignment $\sum_{i=1}^{k} d_H(s, s_i')$.

2. Output an alignment $\mathcal{A}$ such that its consensus cost is minimized among all $\mathcal{A}$'s obtained in step 1(c).

Figure 4.12
PTAS for Average c-Indel Consensus Alignment.

THEOREM 12 (Gusfield 1997) The performance ratio of center star algorithm is 2 for consensus alignment if the cost scheme is metric.

Here we present a PTAS for c-Diagonal Consensus Alignment. The ideas are similar to those of the PTAS for c-Diagonal SP Alignment sketched in the previous section. Again, we need the following restricted version of consensus alignment.

DEFINITION 3 The Average c-Indel Consensus Alignment problem is to find a multiple alignment of $\mathcal{S}$ with the minimum possible consensus cost such that on the average, there are at most c indels per sequence.

The algorithm for this version of consensus alignment is similar to that for Average c-Indel SP Alignment. However, instead of constructing a frequency matrix, here we construct a "center" sequence S. The complete algorithm is given in figure 4.12. Again, the cost scheme assumed here is the simple match/mismatch scheme.

THEOREM 13 For $h > 2$ and $r > 2$, algorithm AverageConsensusAglign produces an alignment with consensus cost at most

$$1 + \max\left\{\frac{4}{h-2}, \frac{8}{\sqrt{e}(\sqrt{4r+1}-3)}\right\}|\Sigma|$$

times that of the optimum in polynomial time, where e is the natural constant.

The proof of the theorem is again quite involved, and we refer to reader to Li et al. (2000) for details. The basic idea of the analysis is to show that the average consensus cost of the $\binom{k}{r}$ different alignments tried in the algorithm is upper bounded by $1 + \max\left\{\frac{4}{h-2}, \frac{8}{\sqrt{e}(\sqrt{4r+1}-3)}\right\}|\Sigma|$ times the optimum. The techniques used here for the analysis of algorithm AverageConsensusAlign are quite different from that for Average SP Alignment.

We now present a PTAS for c-Diagonal Consensus Alignment. The algorithm is almost the same as our PTAS for c-Diagonal SP Alignment: (1) Dynamically cut the k sequences into small segments such that the optimal consensus alignment cost for each group of segments is about ctk for some constant t. This implies that there are at most ct indels in each piece on average. (2) Because of the c-diagonal condition, each cut brings in at most $O(ck)$ errors. Thus, the parameter t acts against the errors caused by the uncertainty of the cutting. (3) Use algorithm AverageConsensusAlign on each group of segments and concatenate the segment alignments together.

The complete algorithm is given in figure 4.13. Here,

$$\rho = 1 + \max\left\{\frac{4}{h-2}, \frac{8}{\sqrt{e}(\sqrt{4r+1}-3)}\right\}|\Sigma|.$$

THEOREM 14 The approximation ratio of algorithm DiagonalConsensusAlign is $\rho\left(1 + \frac{2}{t-2-\frac{1}{c}}\right)$.

The algorithm DiagonalConsensusAlign in fact forms a PTAS for all pairwise cost schemes μ satisfying $\mu(i,i) = 0$ and $\max_{a,b\in\Sigma\cup\{\Delta\}} \mu(a,b)/\min_{a,b\in\Sigma\cup\{\Delta\}} \mu(a,b)$ is bounded by some constant (see Li et al. 2000).

The PTAS's for both the SP model and consensus model can be extended to work for the case where we know the "approximate" positions of all letters of all sequences in an optimal alignment. That is, we know that the j-th letter of s_i is at the position that is at most c positions away (either left or right) from the k-th letter of another sequence, where c is a constant. (This may happen when we are given a multiple

Algorithm DiagonalConsensusAlign

Input: integers c, h, r and t, and $S = \{s_1, \ldots, s_k\}$. Each s_i has length m.

Output: a multiple alignment of S.

1. **for** i from 1 to m **do**

   Let c_i be the cost of the output of algorithm AverageConsensusAlign on input ct, h, r, and $\{s_1[1..i], s_2[1..i], \ldots s_k[1..i]\}$.
2. Let I be the maximum i such that $c_i \leq \rho ctk$. Remove the first I letters from each s_i as a segment.
3. Repeat steps 1, 2, and 3 until each sequence becomes empty.
4. Concatenate the alignments for each group of segments together to obtain a multiple alignment for the original set S.

Figure 4.13
A PTAS for c-Diagonal Consensus Alignment.

alignment whose shape is close to being optimal.) The difference here from c-diagonal alignment is that in an optimal alignment, some sequences may have many spaces clustered in an area because they are not so similar to others in this area. In other words, the spaces are not evenly distributed for some sequences. The extended algorithm is the same except that we may cut sequences at different locations to obtain segments.

4.5.3 Tree Alignment

The first approximation algorithm for tree alignment with guaranteed performance is a ratio-2 algorithm (see Jiang et al. 1994). The algorithm was extended into a polynomial-time approximation scheme in the same paper. In this section, we sketch an improved version of the ratio 2 algorithm (see Wang and Gusfield 1997) and describe briefly how the PTAS works. For convenience, we will use the alternative formulation of tree alignment, that is, we will view tree alignment as a sequence reconstruction problem on trees.

A Ratio-2 Algorithm Using Uniform Lifting Let T be a binary (evolutionary) tree such that each of its leaves is labeled with a unique given sequence. For convenience, we convert T to an *ordered* tree by specifying the children of each internal node as left and right children arbitrarily. A *loaded* tree for T is a tree in which each internal node is also assigned a sequence label (not necessarily equal to a given sequence). A loaded tree is called a *lifted* tree if the sequence label of every internal node v equals the

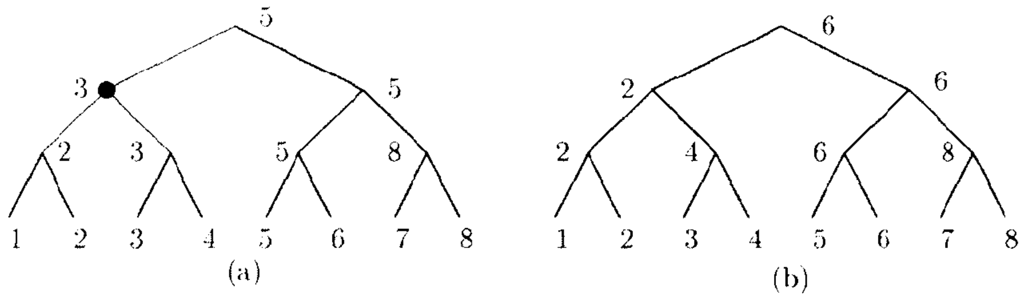


Figure 4.14
(a) A lifted tree. (b) A uniform lifted tree.

sequence label of some child of v. Figure 4.14a exhibits a lifted tree. In the figure, numbers indicate from where a label is lifted. A lifted tree is called a *uniformly* lifted tree if, for each level of T, either every internal node at the level receives its sequence label from its left child or every internal node at the level receives its sequence label from its right child. In other words, the lifting decisions for the internal nodes at the same level are uniform. Figure 4.14b exhibits a uniformly lifted tree.

The following result explains why uniformly lifted trees are interesting.

THEOREM 15 There exists a uniformly lifted tree for T with a cost at most twice the cost of the optimal tree alignment cost.

Now, let us explain how to compute an optimal uniformly lifted tree. To simplify the presentation, we just give an algorithm for full binary trees here, although the algorithm can be easily extended to arbitrary binary trees.

Suppose that T is a full binary tree. Let $V(T)$ denote the set of internal nodes of T and $L(T)$ the set of leaves of T. For each node v, let T_v denote the subtree of T rooted at v and $S(v)$ denote the set of sequence labels of all descendent leaves of v. For each $v \in V(T) \cup L(T)$ and each label $s \in S(v)$, $C[v, s, d]$ denotes the cost of the uniformly lifted tree that labels node v with sequence s. We can compute $C[v, s]$ iteratively using dynamic programming. For each leaf v, we define $C[v, s_i] = 0$ if the label of v is s_i. Let v be an internal node, and v_1 and v_2 its two children. Suppose that $s_i \in S(v_p)$ and $s_j \in S(v_q)$, where $1 \leq p \leq 2$, $q \in \{1, 2\} - \{p\}$, and s_i and s_j are at the same *position* of the subtrees of T_{v_1} and T_{v_2}. (In other words, the two leaves of T_{v_1} and T_{v_2} labeled by the sequences s_i and s_j have the same *rank* in the left-to-right orderings of the leaves in T_{v_1} and T_{v_2}.) Then $C[v, s_i]$ can be computed as follows:

$$C[v, s_i] = C[v_p, s_i] + C[v_q, s_j] + \mu(s_i, s_j) \tag{4.9}$$

where $\mu(s_i, s_j)$ is the optimal pairwise alignment cost between s_i and s_j. Because the sizes of both $V(T) \cup L(T)$ and $S(v)$ are bounded by $O(k)$, we can compute all the

values $C[v, s_i]$ in $O(k^2)$ time if the pairwise alignment costs have been precomputed. Hence the total running time of the algorithm is $O(k^2m^2 + k^2) = O(k^2m^2)$, where m is the length of the given sequences.

In fact, a better bound on the time complexity of the above algorithm can be obtained by a more careful analysis. A pair of two sequences (s_i, s_j) is a *legal pair* if s_i and s_j can be assigned at the ends of a same edge in a uniformly lifted tree. It is easy to see that a sequence s_i can be involved in at most $d(T)$ legal pairs, where $d(T)$ is the depth of T. Thus, there are at most kd legal pairs of sequences in total. Therefore, the running time of our new algorithm is actually $O(kd + kdm^2) = O(kdm^2)$.

Using a data structure called *extended* tree, one can design an algorithm that works for the general binary trees with the same time complexity (Wang and Gusfield 1997).

A PTAS for Tree Alignment Given any lifted tree, we may further reduce its cost by keeping the lifted sequences on some nodes and reconstructing the sequences on the other (internal) nodes to minimize the cost of the edges incident upon these (latter) nodes. For example, based on the lifted sequences 2, 3, 5, we can compute a sequence for the dark circled node in figure 4.14a such that the total cost of the three thin edges incident on the dark circled node is minimized. The new sequence should in general reduce the cost of the tree. This suggests the idea of partitioning a (uniformly) lifted tree into a collection of overlapping components, keeping the lifted sequences at the leaves of these components intact, and optimally reconstructing the sequences for the internal nodes in each component, that is, doing a local optimization on each component. The computation can be done in polynomial time as long as each component has a constant size. Based on this idea, several polynomial time approximation schemes have been proposed (Jiang et al. 1994; Wang and Gusfield 1997; Wang et al. 2001). Wang et al. (2001) gives the fastest algorithm. The running time and performance ratio of the PTAS (Wang et al. 2001) are as follows:

THEOREM 16 For any fixed r, where $r = 2^{t-1} + 1 - q$ and $0 \le q \le 2^{t-2} - 1$, the PTAS runs in time $O(kdm^r)$ and achieves an approximation ratio of $1 + \frac{2^{t-1}}{2^{t-2}(t+1) - q}$. Here the parameter r represents the "size" of local optimization. In particular, when $r = 2^{t-1} + 1$, its approximation ratio is simply $1 + \frac{2}{t+1}$.

4.6 Popular Heuristic Approaches

In this section, we describe some popular heuristic approaches that work reasonably well in practice. Emphasis will be given to the progressive alignment (in particular

Clustal W) and Gibbs sampling paradigms. McClure et al. (1994) offer a comparative study on the performance of some of these methods on protein sequences.

4.6.1 Progressive Alignment Methods

A general strategy of this approach is to progressively merge two multiple alignments of two subsets of sequences into one multiple alignment of the union of the two subsets of sequences. Different progressive alignment methods use different criteria for selecting two subsets of sequences to merge and different algorithms to perform the merge. The best-known progressive alignment programs are perhaps DFALIGN and Clustal W, which merge subsets of sequences and their alignments following a *guide* tree (Feng and Doolittle 1987; Thompson et al. 1994). (Therefore these methods can also be viewed as approximate methods for tree alignment.) In this section, we will only outline the algorithm used in Clustal W. Thompson et al. (1994) offer the details of Clustal W and Feng and Doolittle (1987) for DFALIGN.

The basic algorithm behind Clustal W proceeds in three steps: (1) Compute the optimal alignment cost for each pair of sequences using standard dynamic programming. This results in a *distance matrix* whose entries indicates the degree of divergence of each pair of sequences in evolution. (2) Compute an evolutionary tree from the distance matrix using some phylogeny reconstruction method. This tree will be used as the guide tree. (3) Align the sequences progressively according to the branching order given in the guide tree. The steps are explained in a bit more detail below.

Although the dynamic programming algorithm for pairwise alignment is straightforward, step 1 can in fact be very time consuming and become the bottleneck of the whole process, because here we have to align $\binom{k}{2}$ pairs. So, Clustal W also offers the option of using a fast approximate method based on the exact matching of small tuples (Wilbur and D. Lipman 1983). On the other hand, the (slower) dynamic programming algorithm for pairwise alignment of Clustal W incorporates the notion of *affine gap* cost functions. In a pairwise alignment, a *gap* is defined as a maximal sequence of consecutive spaces. Hence, a gap represents a series of insertions (or deletions) that happened at consecutive positions. Intuitively, it is logical to consider such a series of insertions (or deletions) as a single evolutionary event rather than as independent events, and hence assign cost accordingly. A popular gap cost function, called affine cost function, charges a gap of i spaces with a cost of $g_{open} + i \cdot g_{ext}$, where g_{open} is a constant denoting the cost of opening a gap and g_{ext} is another constant denoting the cost of extending the gap by a space. Pairwise alignment with affine gap costs can be computed in quadratic time by using dynamic programming

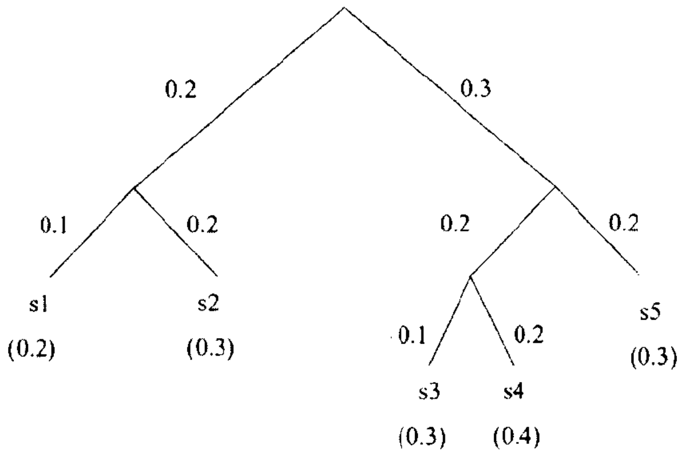


Figure 4.15
Calculating weights for each sequence from the guide tree.

(Gusfield 1997). The pairwise alignment costs are *normalized* taking into account the lengths of sequences involved.

The guide tree is computed from the distance matrix by first using a popular distance-based phylogeny reconstruction method, the Neighbor-Joining method (Saitou and M. Nei 1987). This produces an *unrooted* tree with edge lengths proportional to estimated divergence along each edge. The tree is then converted into a *rooted* tree by placing the root at a "mid-point" on some edge where the means of edge lengths on either side of the mid-point are equal. These edge lengths are also used to derive a *weight* for each sequence as follows: Divide the length of an edge by the number of descendent leaves sharing this edge. The weight of a sequence s is the total (divided) lengths of edges on the path from the root to the leaf labeled by s. Figure 4.15 illustrates an example guide tree with edge lengths and weights for sequences. The sequence weights are given in brackets.

Once a guide tree and sequence weights have been computed, we do progressive alignment by moving from the bottom of the tree toward the root and merging alignments for larger and larger groups of sequences. For example, in the tree given in figure 4.15, we could start by merging the pairwise alignment of sequences s_3 and s_4 with sequence s_5, and then merging this alignment with the pairwise alignment of sequences s_1 and s_2. Each merger involves aligning two multiple alignments, and can be computed by using a dynamic programming algorithm similar to that for the alignment of a pair of sequences, as a multiple alignment can be viewed as a sequence of columns of letters/spaces. In particular, the cost of a pair of columns is calculated as follows. Suppose that $\mathcal{A}_1$ is an alignment of k_1 sequences with weights $w_1, \ldots, w_{k_1}$

and $\mathscr{A}_2$ is an alignment of k_2 sequences with weights $u_1, \ldots, u_{k_2}$. Let $c_1 = (a_1, \ldots, a_{k_1})$ be a column of $\mathscr{A}_1$ and $c_2 = (b_1, \ldots, b_{k_2})$ a column of $\mathscr{A}_2$. Then the cost of these columns is the weighted average of the cost between every letter/space in c_1 and every letter/space in c_2:

$$\mu(c_1, c_2) = \sum_{1 \le i \le k_1, 1 \le j \le k_2} w_i \cdot u_j \cdot \mu(a_i, b_i)/(k_1 k_2)$$

In this way, sequences that are highly divergent from the rest of the sequences are given due consideration in the alignment process. (Note that the cost is in fact similar to SP-cost.)

Dealing with gap costs in the process is a more complicated and subtle issue, and we refer the reader to the original paper (Thompson et al. 1994).

GCG is another software package that is popular in sequence analysis. In GCG, there is a program called PileUp for multiple sequence alignment. The algorithm in PileUp is actually a simplification of DFALIGN and is very similar to earlier versions of Clustal W, although it employs different algorithms to build guide trees and pairwise alignments. We refer the reader to websites such as http://gcg.nhri.org.tw/pileup.html for details.

4.6.2 Iterative Method for Tree Alignment

Sankoff proposed an iterative method for tree alignment (Sankoff et al. 1976; Sankoff and Kruskal 1983). The basic idea is to (1) assign a sequence to each internal nodes, (2) choose an internal node v, and (3) use the three sequences assigned to the three neighbors of v to update the sequence assign to v, by local optimization, and (4) repeat the process until the cost of the tree cannot be improved.

To illustrate the iterative method (Sankoff et al. 1976), consider the phylogeny in figure 4.16, which contains nine species on its leaves. A loaded tree is computed initially (for example, by arbitrarily assigning leaf sequences to internal nodes). To improve the cost of the tree, we divide the phylogeny into seven 3-components, as shown in figure 4.16, each consisting of a center and three terminals. Local optimization is done for every 3-component based on the labels of its three terminals sequentially. The new center label can then be used to update the center label of an overlapping 3-component. The algorithm converges eventually as each local optimization reduces the cost of the tree by at least one. Thus, if the process is repeated long enough, every 3-component will become optimal, although the resulting loaded tree may not be optimal overall. Empirical results show that the algorithm produces a reasonably good loaded tree within five iterations (Sankoff et al. 1976).

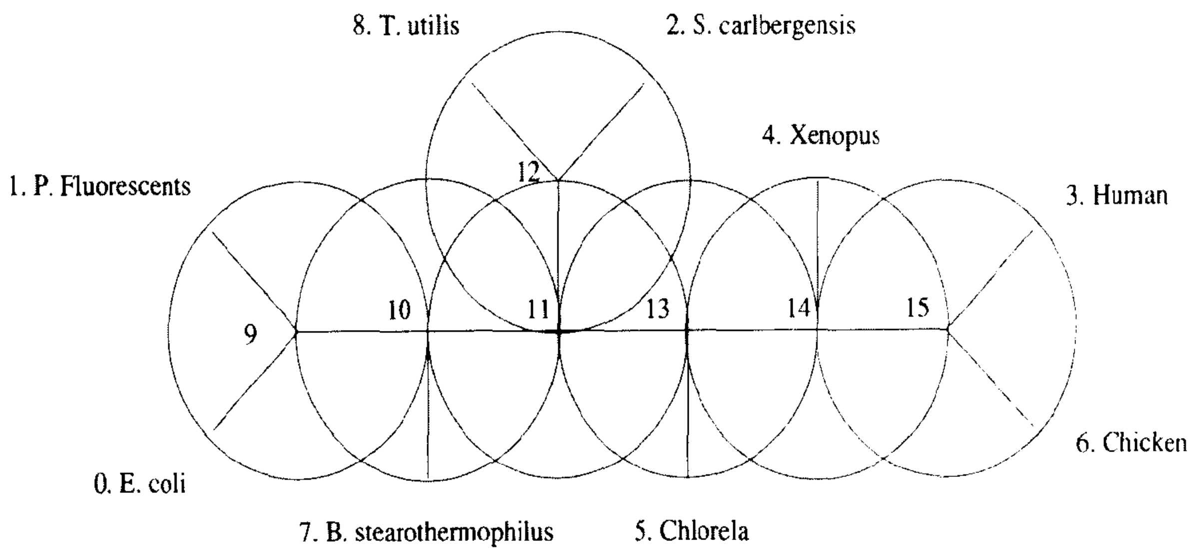


Figure 4.16
A phylogeny with nine species, which is divided into seven 3-components.

4.6.3 Sequence Graph Approach

J. Hein (1989) introduced an approach for tree alignment based on the concept of sequence graph. A sequence graph between two sequences can be obtained from the two-dimensional matrix computed in the dynamic programming algorithm for an optimal pairwise alignment between the two sequences. Recall that the two-dimensional matrix can be computed as follows:

$$c(i, j) = \min\{c(i-1, j-1) + \mu(s_1[i], s_2[j]), c(i-1, j) + \mu(s_1[i], \Delta), \\ c(i, j-1) + \mu(\Delta, s_2[j])\} \quad (4.10)$$

where $c(i, j)$ is the cost between the two prefixess $s_1[1..i]$ and $s_2[1..j]$ and $\mu(a, b)$ is the cost of the pair of opposing letters/spaces a and b. Once the two-dimensional matrix is computed, one can use the standard back-tracing method to obtain an optimal alignment. In the back-tracing process, if value $c(i, j)$ is obtained from $c(i', j')$, where i' (or j') is either i (or j) or $i-1$ (or $j-1$, respectively), then we move from cell (i, j) in the matrix to cell (i', j'). This move determines a column of the optimal alignment. The process is repeated until we reach cell $(0, 0)$ of the matrix. In this way, we compute a path in the matrix that represents an optimal alignment.

In order to obtain all the optimal alignments, we can modify the above back-tracing process such that each time when we move back, we consider *all* the $c(i', j')$'s that lead to the smallest value in equation (4.10). In this way, we obtain a graph from the

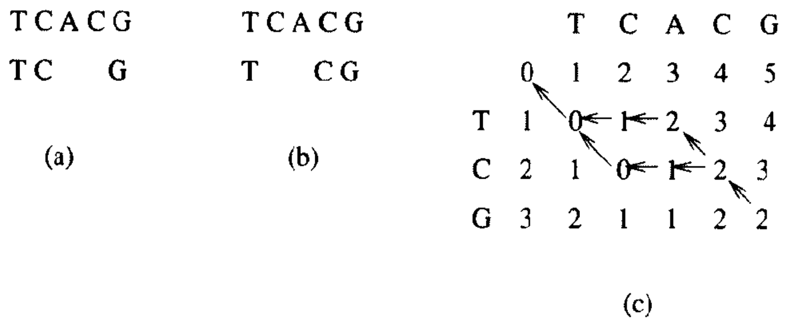


Figure 4.17
(a) An optimal alignment. (b) Another optimal alignment. (c) The sequence graph.

back-tracing process, instead of a path. This graph is called the *sequence graph* between the two sequences, which contains information about all optimal alignments between the two sequences and thus all the "intermediate" sequences between them. In fact, every path in the graph from cell (0, 0) to cell $(|s_1|, |s_2|)$ represents an optimal alignment. An example is given in figure 4.17.

Recall that for tree alignment, we are given a tree T and a set of sequences, one on each leaf of T. Hein's sequence graph method computes a sequence graph between two sequences assigned to a pair of two sibling leaves and assigns the sequence graph on the parent node v. Then it treats v as a leaf (i.e., it deletes the two children of v) and repeats the process until one node remains. In a general step of this process, one may have to align a sequence with a sequence graph or align a sequence graph with another sequence graph. This can be done by using dynamic programming in a way similar to computing a sequence graph between two sequences. Here each node in the sequence graph represents a set of substrings. Let i and j be two nodes in two sequence graphs. $d[i, j]$ denotes the smallest tree alignment cost for any two sequences, one from each sequence graph. The cost $d[i, j]$ can be computed as follows:

$$d[i, j] = \min\{\mu(l(i), l(j)) + d[i', j']\} \tag{4.11}$$

where i' (or j') is either i (or j) or one of the nodes in the sequences graph preceding i (or j), $l(i)$ (or j) is either a space or one of the last letters in the substrings represented by i (or j) (the choices depend on i' [or j']), and the minimum is taken among all possible configurations (see Hein 1989 for details). Eventually, every internal node is assigned a sequence graph. We choose the sequences from the set of sequences represented by sequence graphs that lead to the smallest tree-cost.

This approach is similar to the lifting methods for tree alignment in the sense that lifting methods assign each internal node with a given sequences whereas the sequence graph approach assigns each internal node with a set of intermediate sequences. Note

that this approach cannot guarantee that the solutions obtained are optimal. Moreover, it is not known how well this algorithm approximates the optimum. The time and space complexity of the algorithm in the worst case is exponential in terms of k, the number of given sequences (leaves), because the sequence graph assigned to the root of the tree is in fact k-dimensional.

In practice, this approach works relatively fast and the results obtained are reasonably good (Hein 1989). It is an interesting open problem to give a mathematical analysis on the average size of the sequence graphs used in this approach.

When combined with the progressive alignment approach, the sequence graph method can also be extended to work for SP alignment and consensus alignment.

4.6.4 Stochastic Algorithms

An important application of multiple sequence alignment is in the identification of conserved regions. In this case, we can think of a conserved region as a *pattern* or, as more commonly called, *motif* that appears in multiple sequences or multiple regions of a sequence. Such a motif may represent a significant functional or regulatory element. Recently, there has been extensive research on using stochastic (or probabilistic) algorithms to find motifs based on techniques such as hidden Markov model (HMM) and Gibbs sampling (Durbin et al. 1998). The basic idea behind these algorithms is to treat a motif as a multinomial probabilistic distribution and try to infer the distribution from given data using some kind of unsupervised learning technique. In Gibbs sampling, the elements of a motif are assumed to be more or less independent from each other. On the other hand, HMM relates the elements to each other by the means of a state.

In this section, we only outline how the Gibbs sampling technique works, to give a flavor of the stochastic methods. For details on the HMM technique, we refer the reader to Durbin et al. 1998.

The Motif Identification Problem A motif can be encoded in many ways. For instance, a motif can be encoded as a consensus sequence, an alignment of sequences, or a profile (or frequency matrix), a table giving the probabilities of occurrence of all letters in the sequence-alphabet at each position in the motif (see chapter 14 and Gusfield 1997). To a limited extent, it is possible to transform one type of motif-encoding into another; for instance, a motif encoded as an alignment of sequences can be transformed into a profile by computing the frequencies of occurrence of every letter in each column of the alignment, and this profile can in turn be transformed into a consensus sequence by selecting for each position in that sequence the letter with the maximum frequency of occurrence in that position of the profile.

Each motif has an associated function that is used to assess how well a given sequence matches that motif. A motif matches a sequence if the value of that motif's associated cost (or similarity score) function relative to the motif and that sequence is below (or above) a specified threshold value. The nature of this function depends on the type of motif-encoding; for instance, if the motif is encoded as a sequence, the function might be a distance function between pairs of sequences, and if the motif is encoded as an alignment of sequence, the function might be the cost of an optimal alignment of the sequences already in the alignment and the new sequence. When motifs are encoded as sequences, two popular matching functions are Hamming distance (the number of positions at which letters differ in two sequences of equal length) and edit distance (the minimum number of letter substitutions, insertions, and deletions that must be applied to transform one of the given sequences into the other). Note that both of these functions can be rephrased as similarity functions, which measure the number of identical-letter positions and the maximum number of identical-letter positions relative to a padding of both sequences with special indel letters, respectively. A motif that matches one or more substrings of a given sequence is said to *appear* in that sequence.

A motif may allow *gaps* or spaces, which correspond to positions at which insertions or deletions can occur in the matching of that motif to a given sequences. These gaps can be explicit in the motif itself (either as special letters Δ in a motif-sequence or profile or as the gaps/spaces in an alignment of sequences) or implicit in that motif's associated matching function. For instance, the latter would be the case if a motif is encoded as a sequence and the matching function is edit distance. If the motif incorporates gaps, it is a *gapped motif*; else, it is an *ungapped motif*. Formally, motif identification can be formulated as an optimization problem as:

DEFINITION 4 (Motif Identification) Given a set $\mathcal{S}$ of sequences over alphabet Σ and a motif-to-sequence distance function d, find a motif M and a substring x' (a motif instance) for each sequence $x \in \mathcal{S}$ such that $\sum_{x'} d(M, x')$ is minimized.

Gibbs Sampling for Motif Identification Gibbs sampling is essentially a general stochastic strategy for determining the parameters of a statistical model relative to a given data set. This strategy starts with some setting of parameter-values and iteratively changes the value of one parameter at a time by assuming that the remaining parameters are correct and invoking Bayes's theorem until all parameters converge to stable (if not optimal) values (see Lawrence et al. 1993; Liu et al. 1995, and references for details). With reference to the motif identification problem, the model is a motif encoded as an alignment of sequences, the parameters are the positions of the motif within each sequence in a given set $\mathcal{S}$ (the motif-instances), and the stochastic heuristic

Input: A set $\mathcal{S}$ of sequences over some alphabet.

1. Select initial motif-instances in the sequences of $\mathcal{S}$.

2. Create initial motif from motif-instances.

4. **while** not finished **do**

5. Select sequence s from $\mathcal{S}$.

6. Construct motif from motif-instances in $\mathcal{S} - \{s\}$.

7. Weight all possible motif-instances in s relative to the motif derived above.

8. Stochastically select a new motif-instance x for s relative to these weights.

9. Update motif-instance information for s relative to x.

10. Check if motif has converged and process is finished.

11. Output the current motif and motif-instances.

Figure 4.18
Generic Gibbs sampling algorithm for motif identification.

modifies these motif-instances one sequence at a time, one sequence per iteration, until the alignment of these motif-instances denotes a stable (if not optimal) motif.

Figure 4.18 gives a generic algorithm for Gibbs sampling motif identification. Though several steps of this algorithm may be stochastic, the primary stochastic element is the selection performed in step 8. Under stochastic selection, an element in a set is selected at random relative to the probabilities derived by normalizing the weights assigned to the elements in that set. This type of selection is intuitively more appealing than a deterministic selection that would always select the highest- or lowest-weighted value because stochastic selection can allow a local-search heuristic algorithm to escape from (and hence avoid being trapped in) local optima.

Due to the space constraint, we only describe in more detail how the Gibbs motif identification algorithm works for ungapped motifs. (Extensions to gapped motifs can be found in Lawrence et al. 1993; Liu and Lawrence 1995; Rocke and Tompa 1998.)

Lawrence et al. (1993) give the first algorithm for identifying motifs by Gibbs sampling. The algorithm finds ungapped motifs of a pre-specified length W. In this algorithm, a motif is modeled as a collection of $W + 1$ multinomial probability distributions over the sequence alphabet Σ, where the first W of these distributions correspond to a profile-encoding of the motif, that is, the first W of these distributions

correspond to the frequencies of occurrence $q_{i,j}$, $1 \leq i \leq W$, and $1 \leq j \leq |\Sigma|$, of letter j at position i in the motif, and the final distribution corresponds to the "background" frequencies of occurrence p_j, $1 \leq j \leq |\Sigma|$, of letter j in parts of the sequences that are not in the motif. A candidate motif-instance sequence $x = x_1 x_2 \ldots x_W$ over alphabet Σ is evaluated against the motif in terms of the ratio

$$Q_x/P_x = \prod_{i=1}^{W} q_{i,\mathbf{sym_ind}(x_i)}/p_{\mathbf{sym_ind}(x_i)} \tag{4.12}$$

where $\mathbf{sym_ind}(s)$ is the index of letter s in Σ. For numerical reasons, this product of ratios is more often computed as an equivalent sum of log-ratios.

The so called *F-value* is defined (Lawrence et al. 1993) to measure the quality of a motif

$$F = \sum_{j=1}^{W} \sum_{i=1}^{|\Sigma|} c_{i,j} \log q_{i,j}/p_j \tag{4.13}$$

where $c_{i,j}$ is the unnormalized count of the number of occurrences of letter i at position j in the motif, and $q_{i,j}$ and p_j are computed from the motif in a way similar to equation 4.12.

Intuitively, by seeking motifs that maximize the ratio Q_x/P_x, the algorithm is searching for the motif whose collective letter-occurrence distribution is probabilistically the most distinct from the background letter-occurrence distribution. As such, the distance function encoded in this algorithm is a variant of the Kullback-Leibler distance

$$H(Q \,||\, P) = \sum_x Q(x) \log Q(x)/P(x) \tag{4.14}$$

that gives a measure of the distinctness of probability distributions Q and P. ($H[Q \,||\, P]$ is also known as the *relative entropy* of Q to P). This connection is more easily seen in the re-formulation of the ratio Q_x/P_x in terms of F.

Each iteration of the main loop in this algorithm runs in $O(km + W|\Sigma|)$ space and $O(kmW|\Sigma|)$ time, where k is the number of given sequences and m is the length of the longest given sequence. Simulations and rules-of-thumb derived from practical experience (Lawrence et al. 1993) suggest that the number of iterations required for the algorithms to converge on a motif is small on real datasets; however, no upper bounds on the number of iterations is known, and hence no worst-case time complexity can be given for the algorithm as a whole. That being said, it is known that if

the algorithm is allowed to run to infinity, it will always find an optimal motif (Liu et al. 1995; Liu and Lawrence 1995).

The algorithm described above is the most basic version of the Gibbs sampling motif-finding algorithm, in that it assumes that there is one motif in the given set of sequences, one copy of that motif in each sequence, and the length of that motif is known. Modifications of this basic algorithm that allow it to automatically set motif length and automatically determine both the number of copies of a motif in each sequence as well as the number of motifs present in the set of given sequences are described (albeit often in statistical rather than algorithmic terms) in Lawrence et al. 1993; Liu and Lawrence 1995; and Liu et al. 1995.

4.7 Concluding Remarks and Open Problems

Although many approaches have been attempted for multiple sequence alignment and many algorithms designed, the problem still remains one of the most challenging problems in computational biology, both theoretically and practically.

A major open problem in theory is if SP alignment has a PTAS assuming metric pairwise cost schemes. Open questions that are more relevant in practice include (1) How do we make the multiple alignment programs fast enough to handle hundreds or thousands of sequences simultaneously? (2) How do we integrate multiple sequence alignment with phylogeny reconstruction? And (3) Can we make multiple sequence alignment a more interactive process to take advantage of special domain knowlege about the sequences being aligned? Question (2) is closely related to the generalized tree alignment problem that is known to be MAX SNP-hard and thus have no PTAS. Question (3) could be a key for overcoming the combinatorial complexity of multiple sequence alignment.

Acknowledgments

Tao Jiang is supported in part by NSERC Research Grant OGP0046613, a UCR startup grant and NSF grants CCR-9988353 and ITR-0085910. Lusheng Wang is supported in part by Hong Kong CERG Grants 9040297, 9040352, 9040444, and CityU Strategic Grant 7001025.

References

Altschul, S., and Lipman. D. (1989). Trees, stars, and multiple sequence alignment. *SIAM Journal on Applied Math.* 49: 197–209.

Apostolico, A., and Giancarlo, R. (1998). Sequence alignment in molecular biology. *Journal of Computational Biology* 5: 173–198.

Baconn, D., and Anderson, W. (1986). Multiple sequence alignment. *Journal of Molecular Biology* 191: 153–161.

Bafna, V., Lawer, E., and Pevzner, P. (1997). Approximation algorithms for multiple sequence alignment. *Theoretical Computer Science* 182: 233–244.

Bonizzoni, P., and Della Vedova, G. The complexity of multiple sequence alignment with SP-score that is metric. *Theoretical Computer Science*, to appear.

Carrillo, H., and Lipman, D. (1988). The multiple sequence alignment problem in biology. *SIAM Journal on Applied Math* 48: 1073–1082.

Chan, S. C., Wong, A. K. C., and Chiu, D. K. T. (1992). A survey of multiple sequence comparison methods. *Bulletin of Mathematical Biology* 54: 563–598.

Chao, K., Pearson, W. R., and Miller, W. (1992). Aligning two sequences within a specified diagonal band. *CABIOS* 8: 481–487.

Durbin, R., Eddy, S., Krogh, A., and Mitchison, G. (1998). *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids.* New York: Cambridge University Press.

Feng, D., and Doolittle, R. F. (1987). Progressive sequence alignment as a prerequisite to correct phylogenetic trees. *J. Mol. Evol.* 25: 351–360.

Fickett, J. W. (1984). Fast optimal alignment. *Nucl. Acids Res.* 12: 175–180.

Galil, Z., and Ciancarlo, R. (1989). Speeding up dynamic programming with applications to molecular biology. *Theoretical Computer Science* 64: 107–118.

Garey, M. R., and Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-completeness.* San Francisco: W. H. Freeman.

Gupta, S., Kececioglu, J., and Schaffer, A. (1995). Making the shortest-paths approach to sum-of-pairs multiple sequence alignment more space efficient in practice. In *Proc. 6th Symp. on Combinatorial Pattern Matching, LNCS937*, 128–143. Berlin: Springer.

Gusfield, D. (1997). *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology.* New York: Cambridge University Press.

Gusfield, D. (1993). Efficient methods for multiple sequence alignment with guaranteed error bounds. *Bulletin of Mathematical Biology* 55: 141–154.

Hein, J. (1989a). A new method that simultaneously aligns and reconstructs ancestral sequences for any number of homologous sequences, when the phylogeny is given. *Mol. Biol. Evol.* 6: 649–668.

Hein, J. (1989b). A tree reconstruction method that is economical in the number of pairwise comparisons used. *Mol. Biol. Evol.* 6: 669–684.

Hein, J. (1990). Unified approach to alignment and phylogenies. *Methods in Enzymology* 183: 626–645.

Henikoff, S., and Henikoff, J. G. (1992). Amino acid substitution matrices from protein blocks. *Proc. Natl. Acad. Sci. U.S.A.* 89: 10915–10919.

Hochbaum, D. (1996). *Approximation Algorithms for NP-hard Problems*, PWS publishers.

Jiang, T., Kearney, P., and Li, M. (2000). Open problems in computational molecular biology. *Journal of Algorithms* 34-1: 194–201; also *SIGACT News* 30-3: 43–49, 1999.

Jiang, T., Lawler, E. L., and Wang, L. (1994). Aligning sequences via an evolutionary tree: Complexity and approximation. In *Proc. 26th ACM Symp. on Theory of Computing* 760–769. New York: ACM Press.

Just. W. (1998). On the computational complexity of gap-0 multiple alignment. Manuscript.

Lawrence, C. E., Altschul, S. F., Boguski, M. S., Liu, J. S., Neuwald, A. F., and Wooton, J. C. (1993). Detecting subtle sequence signals: A Gibbs sampling strategy for multiple alignment. *Science* 262: 208–214.

Li, M., Ma, B., and Wang, L. (1999). Finding similar regions in many strings. In *Proc. 31st ACM Symp. on Theory of Computing* 473–482. New York: ACM Press.

Li, M., Ma, B., and Wang, L. (2000). Near optimal multiple alignment within a band in polynomial time. In *Proc. 32nd ACM Symp. on Theory of Computing*, 425–434.

Lipman, D., Altschul, S. F., and Kececioglu, J. D. (1989). A tool for multiple sequence alignment. *Proc. Natl. Acad. Sci. USA* 86: 4412–4415.

Liu, J. S., and Lawrence, C. E. (1995). Statistical models for multiple sequence alignment: Unifications and generalizations. In *American Statistical Association: Proceedings of the Statistical Computing Section*, 1–8.

Liu, J. S., Neuwald, A. F., and Lawrence, C. E. (1995). Bayesian models for multiple local sequence alignment and Gibbs sampling strategies. *J. Am. Statist. Assoc.* 90(432): 1156–1170.

McClure, M., Vasi, T., and Fitch, W. (1994). Comparative analysis of multiple protein-sequence alignment methods. *Mol. Biol. Evol.* 11(4): 571–592.

Myers, G., Selznick, S., Zhang, Z., and Miller, W. (1997). Progressive multiple alignment with constraints. In *Proceedings of the First Annual International Conference on Computational Molecular Biology*, 220–225.

Pearson, W. R. (1990). Rapid and sensitive comparison with FASTP and FASTA. *Methods Enzymol.* 183: 63–98.

Pearson, W. R. (1991). Searching protein sequence libraries: Comparison of the sensitivity and selectivity of the Smith-Waterman and FASTA algorithms. *Genomics* 11: 635–650.

Pearson, W. R., and Lipman, D. (1988). Improved tools for biological sequence comparison. *Proc. Natl. Acad. Sci. USA.* 85: 2444–2448.

Pevzner, P. (1992). Multiple alignment, communication cost, and graph matching. *SIAM Journal on Applied Math* 56: 1763–1779.

Ravi, R., and Kececioglu, J. (1995). Approximation algorithms for multiple sequence alignment under a fixed evolutionary tree. In *Proc. 6th Symp. on Combinatorial Pattern Matching*. *LNCS937*, 330–339. Berlin: Springer.

Riordan, J. R., Rommens, J. M., Kerem, B., Alon, N., Rozmahel, R., Grzelczak, Z., Zielenski, J., Lok, S., Plavsic, N., Chou, J., Drumm, M. L., Iannuzzi, M. C., Collins, F. S., and Tsui, L. (1998). Identification of the cystic fibrosis gene: Cloning and characterization of complementary DNA. *Science* 245: 1066–1073.

Rocke, E., and Tompa, M. (1998). An algorithm for finding novel gapped motifs in DNA sequences. In *Proceedings of The Second Annual International Conference on Computational Molecular Biology (RECOMB'98)*, 228–233. New York: ACM Press.

Saitou, N., and Nei, M. (1987). The neighbor-joining method: A new method for reconstructing phylogenetic trees. *Molecular Biology and Evolution* 4: 406–425.

Sankoff, D. (1975). Minimal mutation trees of sequences. *SIAM Journal on Applied Math* 28: 35–42.

Sankoff, D., Cedergren, R. J., and Lapalme, G. (1976). Frequency of insertion-deletion, transversion, and transition in the evolution of 5S ribosomal RNA. *J. Mol. Evol.* 7: 133–149.

Sankoff, D., and Kruskal, J. (1983). Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison. Reading, MA: Addison Wesley.

Sankoff, D., and Rousseau, P. (1975). Locating the vertices of a Steiner tree in an arbitrary metric space. *Math. Program.* 9: 240–246.

Schwikowski, B., and Vingron, M. (1997). The deferred path heuristic for the generalized tree alignment problem. In *Proceedings of the first annual International Conference on Computational Molecular Biology (RECOMB'97)*, 257–266. New York: ACM Press.

Schuler, G. D., Altschul, S. F., and Lipman, D. J. (1991). A workbench for multiple alignment construction and analysis. In *Proteins: Structure, Function and Genetics* 9: 180–190.

Schwarz, R., and Dayhoff, M. (1979). Matrices for detecting distant relationships. In *Atlas of Protein Sequences*, Dayhoff, M. ed., National Biomedical Research Foundation. 353–358.

Spouge, J. L. (1991). Fast optimal alignment. *CABIOS* 7: 1–7.

Thompson, J., Higgins, D., and Gibson, T. (1994). CLUSTAL W: Improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucl. Acids Res.* 22(22): 4673–4680.

Ukkonen, E. (1985). Algorithms for approximate string matching. *Inform. Control* 64: 100–118.

Wang, L., and Jiang, T. (1994). On the complexity of multiple sequence alignment. *J. Comput. Biol.* 1: 337–348.

Wang, L., Jiang, T., and Lawler, E. L. (1996). Approximation algorithms for tree alignment with a given phylogeny. *Algorithmica* 16: 302–315.

Wang, L., and Gusfield, D. (1997). Improved approximation algorithms for tree alignment. *J. Algorithms* 25: 255–173.

Wang, L., Jiang, T., and Gusfield, D. (2001). A more efficient approximation scheme for tree alignment. *SIAM J. Comput.* 30: 283–299.

Wang, L., Ma, B., and Li, M. (2000). Fixed topology alignment with recombination. *Discrete Appl. Math.* 104: 281–300.

Waterman, M. S. (1989). Sequence alignments. In *Mathematical Methods for DNA Sequences*, Waterman M. S. ed., Boca Raton, FL: CRC, 53–92.

Waterman, M. S. (1995). *Introduction to Computational Biology: Maps, Sequences, and Genomes*. London: Chapman and Hall.

Wilbur, W. J., and Lipman, D. (1983). Rapid similarity searches of nucleic acid and protein data banks. *Proc. Natl. Acad. Sci. USA* 80: 726–730.

5 Phylogenetics and the Quartet Method

Paul Kearney

5.1 Introduction

Evolution is a catchall phrase that encompasses a collection of processes that operate on DNA sequences. Roughly speaking, these processes operate on the species, genome and, nucleotide levels.

A species is a population of individuals in which similar yet distinct genotypes are observed. Genotypes differ due to gene variants called alleles. Allele frequencies within a species change from generation to generation due to forces such as random genetic drift and natural selection.

Genomes are also altered by events such as gene duplication, horizontal gene transfer, gene rearrangements, and tandem sequence duplications. These events can act as forces of innovation. For example, a gene duplication event can yield two genes, A and B, that initially have the same function. However, over time they may follow different evolutionary paths, permitting B to take on a function distinct from the function of A. An example of this is trypsin and chromotrypsin (Barker and Dayhoff 1980), which cleave polypeptide chains at different residues.

Finally, evolution also operates on a very low level where nucleotides are substituted, inserted, and deleted. These point mutations can be advantageous, deleterious, or neutral in their effect upon genes. To illustrate by analogy, consider the three mutations of the phrase "nothing last forever" below and the effect that the mutations have upon its meaning.

no thing last forever

nothing lass forever

nothing lasts forever

The first mutation (insertion of a space) is neutral in the sense that the meaning of the phrase is still clear. The second mutation (substitution of "s" for "t") is deleterious because the phrase is now meaningless. The third mutation (insertion of "s") is advantageous in the sense that grammatically, the sentence has been improved. The same results hold true for gene sequences. Point mutations can have little functional effect upon the gene, cause the gene to lose function, or improve the functionality of the gene.

Computational techniques are employed in the analysis of all evolutionary processes, the details of which would fill several texts. This chapter focuses on the development of computational techniques for the analysis of gene sequence evolution by point mutation. Gene sequence evolution, and more generally molecular evolu-

tion, are of interest for several reasons:

- Because gene sequences are the direct product of evolution, gene sequences contain clues to the evolutionary processes that produced them.
- Gene sequences encode proteins that are the functional and structural units of life. Studying the evolution of gene sequences permits an understanding of how biological functions have evolved.
- Unlike morphological features, there are genes that are shared by most organisms. Consequently, evolutionary studies of species using gene sequences can permit more breadth than the evolutionary study of species using morphological features.
- In 1963, Margoliash published the first evolutionary tree based on the amino acid sequence of the protein cytochrome c (Margoliash 1963). Advances in sequencing technology now permit the routine evolutionary analysis of large collections of gene and protein sequences.[1]

This chapter avoids, when possible, material covered in excellent sources such as *Molecular Phylogenetics* (Swofford et al. 1996) and *Molecular Evolution* (Li 1997). These books discuss the application of computational techniques to sequence data, whereas this chapter focuses on the development of computational techniques for the evolutionary analyses of gene sequences. To this end, we will use the quartet method as an illustrative example. In section 5.3, we present an introduction to the quartet method and its foundations. In section 5.4, we offer and assess several examples of the quartet method. Finally, in section 5.5 we discuss future trends and present a list of resources.

5.2 Rational Development of Computational Methods for Evolutionary Analyses

Phylogenetics is the design and development of computational and statistical methods for evolutionary analyses. The general concepts that arise in phylogenetics are briefly introduced here and will be discussed in more depth in later sections. For the purposes of this discussion, we restrict our attention to the evolutionary analyses of sequences.

5.2.1 Models of Evolution

The rational development of a phylogenetic method requires a model of evolution as a starting point. Models of evolution have two components: a model of cladogenesis and a model of gene sequence evolution.

1. For example, the Ribosomal Database Project contains evolutionary trees based on ribosomal RNA sequences obtained from thousands's of species (Maidak et al. 1999).

Models of cladogenesis describe the process of species/sequence creation and species/sequence loss (Slowinski and Guyer 1989; Losos and Adler 1995). Fundamental to these models is the belief that the evolutionary history of a family of related genes can be represented by an evolutionary tree:

DEFINITION 1 Let S be a set of gene sequences. An evolutionary tree T for S is a rooted tree where the leaves are labeled by elements of S. T is called weighted if the edges of T have associated lengths.

Internal nodes of an evolutionary tree represent ancestral gene sequences. Consequently, internal nodes also correspond to speciation events where the parent lineage gives rise to two or more child lineages. If an evolutionary tree is binary (all internal nodes have two children), it is called *resolved.* Edge weights are typically proportional to the amount of mutation that has occurred from the parent sequence to the child sequence.

One must be careful to distinguish between species trees and gene trees. A species tree represents the evolutionary history of a collection of species, whereas a gene tree represents the evolutionary history of a collection of related genes. Even if homologous genes are sampled from each species, the resulting gene tree may not be the same as the species tree (Li 1997) due to paralogous and orthologous gene evolution. Even the assumption that evolutionary histories can be described using trees is not always true due to horizontally transferred genes (Li 1997).

Models of sequence evolution, such as the Jukes-Cantor model (Jukes and Cantor 1969), describe how sequences evolve over time as a result of point mutations. Models are varied and incorporate transition versus transversion bias, rate variance among sites, codon position bias, and other complexities of sequence evolution. These models are stochastic in nature. The reader is directed to Swofford et al. 1996 for a detailed presentation. Note that these models do not address the evolutionary of protein secondary and tertiary structure constraints (Benner et al. 1997, 1993).

5.2.2 Methodology

Phylogenetic methods vary considerably in the concepts upon which they are developed. In this section, we briefly overview and contrast several popular phylogenetic methods, which are discussed in detail elsewhere (Swofford et al. 1996; Li 1997). In the next section, we explore the quartet method in depth.

The maximum likelihood method (Felsenstein 1981) is widely accepted as the most accurate method for inferring evolutionary trees from sequence data. This statistical approach is based on an assumed model of evolution. The goal is to obtain the evolutionary tree that most likely produced the observed sequences. However, the

maximum likelihood tree is extremely costly to compute as it requires a search of the entire space of evolutionary trees, including ancestral sequence assignments. Heuristic versions of maximum likelihood have been developed, such as fastDNAML (Olsen et al. 1994), which can analyze large data sets but at the cost of accuracy.

The maximum parsimony method (Farris 1970) is based on the assumption that the correct evolutionary tree is the one that requires the smallest number of point mutations to explain the observed sequences. Of course, this assumption is violated in reality, as point mutations can be superimposed; nevertheless, for low mutation rates, the assumption is reasonable. The maximum parsimony method lends itself to discrete analysis, and so has received much attention from the computational biology research community.

Distance methods such as neighbor joining (Saitou and Nei 1987) are based on the observation that the distance metric d_T of an evolutionary tree T is unique to T (Hakimi and Yau 1964; Waterman et al. 1977) where, for each pair of sequences x and y, $d_T(x, y)$ is the path length in T from x to y. Because d_T is unique to T, d_T is very specific and useful information for reconstructing T. Distance methods estimate $d_T(x, y)$ by assessing the similarity of sequences x and y and then correcting for unobserved superimposed mutations under some model of sequence evolution. This information is then used to produce an estimate of T. A criticism of distance methods is that they lose information by reducing sequence information to similarity data, whereas maximum likelihood and maximum parsimony do not. However, distance methods tend to be very efficient, permitting the analysis of large data sets.

Many phylogenetics methods, typically distance methods, require an initial alignment of the sequences. In this case, the alignment of the sequences is itself an evolutionary hypothesis and affects the accuracy of the evolutionary analysis (Doolittle 1986; Feng and Doolittle 1987). Consequently, multiple sequence alignment methods are phylogenetic methods. Recently, distance methods that do not require sequence alignment have been developed (Li et al. 2001).

5.2.3 Assessment

There are many criteria by which a phylogenetic method can be assessed. Some of these are discussed briefly below. More detailed discussions appear in the following section.

Topological accuracy The topology of an evolutionary tree is defined by its set of edges. An edge e of an evolutionary tree T is defined by the bipartition (X, Y) of the sequences induced by the removal of e from T, yielding two evolutionary trees labeled by X and Y, respectively. Given two evolutionary trees T_1 and T_2, their topological

intersection is the set of edges common to T_1 and T_2. Clearly, the larger the topological intersection between T and an estimate T' of T, the better the estimate.

Consistency A phylogenetic method is consistent if it converges to the correct evolutionary tree given an infinite amount of sequence data (Felsenstein 1988). In practice there is not an infinite, or even a large, amount of sequence data. Nevertheless, consistency is considered a desirable property, whereas inconsistency (Felsenstein 1978a) is an indication that the phylogenetic method has limitations.

Power The power of a consistent phylogenetic method is the rate at which it converges to the correct evolutionary tree as more and more sequence data is used. More powerful methods require less sequence data to produce the correct evolutionary tree, and so are preferable.

Computational efficiency The problem of inferring an evolutionary tree is large on many dimensions. In particular, the number of evolutionary trees with n leaves is enormous:

$$\frac{(2n-3)!}{2^{n-2}(n-2)!}$$

As data sets become larger and larger, the efficiency of a phylogenetic method becomes crucial. In particular, present-day datasets are so large that only heuristic methods can be used for analysis.

Robustness A phylogenetic method is robust if it remains accurate even when its assumptions are violated. This is important because phylogenetic methods are based upon simplified models of evolution and are used to analyze sequence data sets that have evolved in different ways.

Transparency The reality is that many biologists will only use phylogenetic methods that are understandable, and so often mathematically complicated methods are not utilized. In particular, methods that make their biological assumptions explicit are preferable.

5.3 Introduction to Quartet Methods

The quartet method is a paradigm for developing phylogenetic methods. In this section, I introduce the quartet method and explore its foundations. In the next section, I present several examples of the quartet method. A diagram depicting the two stages of the quartet method appears in figure 5.1.

The input to the quartet method is a collection of sequences S. A *quartet* is a set of four sequences, and a *quartet topology* is an evolutionary tree for a set of four

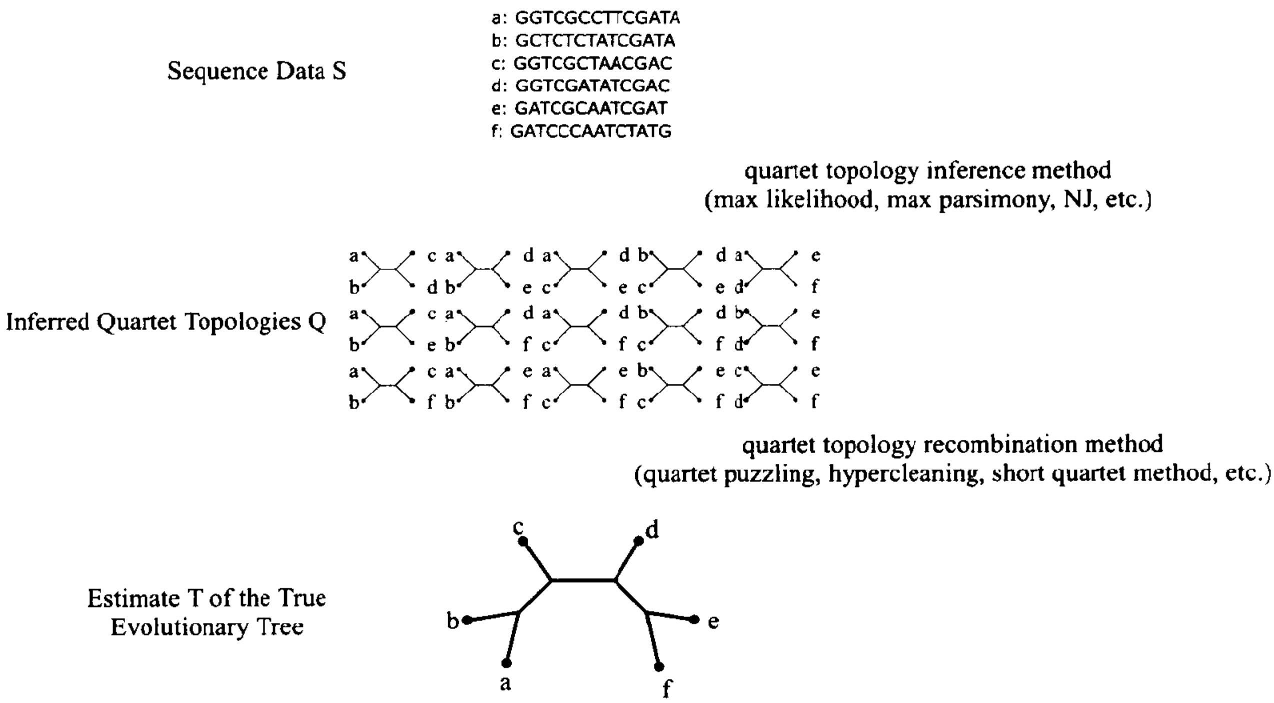


Figure 5.1
The two stages of the quartet method.

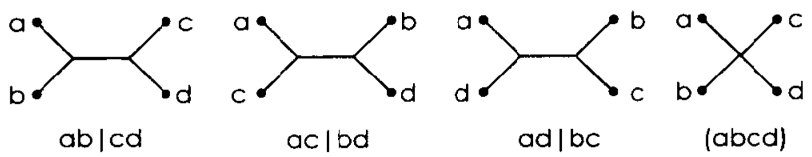


Figure 5.2
The four possible quartet topologies for quartet $\{a, b, c, d\}$.

sequences. The first stage of the quartet method infers a set Q of quartet topologies from S using a phylogenetic method such as maximum likelihood, maximum parsimony, or neighbor joining. The four possible quartet topologies for sequence quartet $\{a, b, c, d\}$, denoted $ab \mid cd$, $ac \mid bd$, $ad \mid bc$ and $(abcd)$, appear in figure 5.2. The first three of these quartet topologies are resolved, whereas the last is unresolved. Quartet topology $ab \mid cd$ is induced in an evolutionary tree T if $P_T(a, b) \cap P_T(c, d) = \emptyset$.

The second stage of the quartet method is called *recombination*. In this stage, the quartet topologies in Q are recombined to form an estimate T of the unknown evolutionary tree T_{true}, where T_{true} is the evolutionary tree that models the actual historical evolution of sequences in S. This requires the definition of an optimization criterion for assessing an evolutionary tree T, given Q and an algorithm for obtaining

or approximating the optimal evolutionary tree. The computational challenge and biological difficulty of recombination is that Q will contain quartet errors that make the recombination stage nontrivial. A quartet topology $ab \mid cd \in Q$ is a quartet error if $ab \mid cd \notin Q_{T_{true}}$, where Q_T denotes the set of quartet topologies induced in an evolutionary tree T. That is, Q is an estimate of $Q_{T_{true}}$.

When stated explicitly, the optimization criterion upon which current quartet methods are based is typically a variation of the following:

MAXIMUM QUARTET CONSISTENCY (MQC)
Instance: A set Q of quartet topologies over sequence set S.
Goal: Find an evolutionary tree T for sequences S that maximizes $|Q_T \cap Q|$.

There are many variations of MQC. For example, quartet topologies may be assigned confidence values or weights that can be incorporated into the recombination optimization criterion. A set of quartet topologies is *interweighted* if each quartet topology q is assigned a nonnegative weight $w(q)$. A set of quartet topologies is *intraweighted* if for each quartet $\{a, b, c, d\}$, weights for $ab \mid cd$, $ac \mid bd$, and $ad \mid bc$ are specified. If a set of quartet topologies Q contains a quartet topology for each quartet of sequences in S, then Q is called *complete*. Otherwise, Q is *incomplete*. A set of quartet topologies is *rooted* if each quartet topology is assigned a root. These roots can then be utilized during recombination.

Like many phylogenetic methods, some quartet methods produce an unrooted and unweighted evolutionary tree T. Other methods can then be applied to obtain edge weights if desired (Swofford et al. 1996). Usually the outgroup method is used to determine the root of T. For the remainder of the chapter, one can assume that the evolutionary trees discussed are unrooted and unweighted.

5.3.1 Foundations of the Quartet Method

Several requirements must be met in order for the quartet method to be a viable phylogenetic method:

- Q_T must contain sufficient information to reconstruct T. In fact, because the set Q of quartet topologies inferred from the sequence data S almost always contains quartet errors, it is necessary that estimates Q of Q_T be sufficient to reconstruct T.

- Quartet methods must either accelerate existing methods such as maximum likelihood and maximum parsimony, or improve their accuracy. That is, it must be either more efficient or more accurate to use a quartet method to obtain an estimate, where the set Q of quartet topologies is obtained using maximum likelihood, than it is to obtain an estimate directly from the sequence data using maximum likelihood.

• Q must be a good estimate of Q_T, otherwise it would be impossible to reconstruct T from Q.

Let's review evidence that the quartet method satisfies these requirements. The mathematical basis for the quartet method is formalized by the four point condition (Buneman 1971):

THEOREM 1 D is the distance metric of an evolutionary tree if and only if, for all quartets $\{a, b, c, d\}$:

$$D(a, b) + D(c, d) \leq D(a, c) + D(b, d) = D(a, d) + D(b, c)$$

for some permutation of a, b, c, and d, where $D(x, y)$ denotes the path length from leaf x to leaf y.

It follows from the four-point condition that indeed quartet topology information is highly specific and can be used to recover an evolutionary tree (Colonius and Schulze 1981):

THEOREM 2 Q_T is unique to T and T can be reconstructed from Q_T efficiently.

A more relevant result for the recovery of an evolutionary tree T from an estimate Q of Q_T, as is necessary in practice, appears below (Jiang et al. 1998). Here an edge e of an evolutionary tree T is defined by the bipartition (X, Y) of the sequences induced by the removal of e from T yielding two evolutionary trees labeled by X and Y, respectively. Given Q and an edge $e = (X, Y)$, the number of quartet errors on e is

$$|\{xx' \mid yy' \mid x, x' \in X, y, y' \in Y\} - Q|$$

THEOREM 3 If each edge $e = (X, Y)$ of T has less than $(|X| - 1)(|Y| - 1)/2$ quartet errors, then T is the unique evolutionary tree that minimizes $|Q_T - Q|$. Furthermore, this bound on quartet error is tight.

This result quantifies the amount of quartet error that is tolerable while Q still gives specific information about T. In section 5.4.5, the hypercleaning algorithm is presented, which realizes T given a set Q satisfying the quartet error bound in theorem 3.

The computational basis for the quartet method is that computationally intensive methods such as maximum likelihood and maximum parsimony, though infeasible for inferring even moderately sized evolutionary trees, can be applied to infer quartet topologies efficiently. A discussion of the tractability of recombination appears be-

low, in section 5.4.2. Initial research indicates that the quartet method is more efficient than methods such as maximum parsimony and maximum likelihood (Berry et al. 2000, 1999; Strimmer and von Haeseler 1996), while still providing a high degree of accuracy. However, further research is required.

5.3.2 Inferring Quartet Topologies and Taxonomic Sampling

Whereas quartet recombination is primarily the combinatorial problem of reassembling pieces (quartet topologies) of the unknown evolutionary tree, the inference stage is primarily a biological problem where the goal is to infer the evolutionary history of a sequence quartet. There is evidence to suggest that this dichotomy has advantages:

- General advances in phylogenetic methods automatically transfer to the quartet method. Specifically, phylogenetic methods can be optimized for inferring quartet topologies. For example, maximum likelihood methods such as fastDNAML (Olsen et al. 1994) are heuristic for moderate to large sequence sets but are exact methods for instances of size four.
- Our knowledge of how well various phylogenetic methods infer quartet topology under various conditions is substantial (for example, see Huelsenbeck and Hillis 1993). This wealth of knowledge can be leveraged to infer quartet topology more accurately.

The ability to accurately infer quartet topologies is closely related to the concept of taxonomic sampling. Hendy and Penny (1989) introduced the idea that adding taxa (in our case sequences) to the dataset so that long branches of the evolutionary tree are shortened may increase the accuracy of the resulting estimate (see also Lecointre et al. 1993). This idea later received support from a study conducted by Hillis (1996) that resulted in a series of papers validating, criticizing, or clarifying taxonomic sampling (e.g., Kim 1996; Greybeal 1998; Poe 1998; Smith and Warnow 1998).

From the perspective of the quartet method, the relevant question is whether or not quartet topologies are more accurately estimated when embedded within a larger set of sequences. That is, should a given quartet topology be extracted from an evolutionary tree inferred from a super sample of the quartet of interest?

Graybeal (1998) used a simulation study to examine the effect of adding taxa to a Felsenstein zone (Felsenstein 1978b) quartet topology and found that there was an advantage to super sampling a quartet topology when using maximum parsimony but not when using maximum likelihood. Similarly, Smith and Warnow (1998) used a simulation study that examined the effects of adding more taxa to a Felsenstein

zone quartet topology and found that when sequence length is sufficiently long, there is an advantage to super sampling when using maximum parsimony and neighbor joining.

Although these results suggest that super sampling a single pathological quartet results in improved accuracy, they do not address the issue of how the accuracy of all quartets are affected by super sampling. For example, super sampling can create additional Felsenstein zone quartet toplogies. Badger and Kearney (2000) examined the simultaneous effects of super sampling on distributions of quartet topologies in an experimental study where it was found that the overall accuracy of quartets was not significantly increased by super sampling strategies.

5.4 A Survey of Quartet Methods

Quartet methods have received much attention in both the biological and computational communities in recent years. Here some of the highlights of this research is presented. Although this survey is certainly not exhaustive, it does present several important and interesting concepts that arise in the development of quartet methods. Essential contributions of the presented methods are presented, along with critical assessments and indications of research remaining to be conducted.

5.4.1 Quartet Puzzling

Quartet puzzling, introduced by Strimmer and von Haeseler (1996), is designed to be a practical heuristic quartet method for inferring evolutionary trees and is currently the most widely used quartet method.

Method Overview Quartet puzzling proceeds by first inferring quartet topologies using maximum likelihood, although any phylogenetic method could be used. The sequences are then randomly ordered and an evolutionary tree is built by sequentially inserting the sequences into the evolutionary tree. The branch of the evolutionary tree onto which sequence s is inserted is determined by polling all quartets involving sequence s. Specifically, for each quartet topology $sx \mid yz$, the edges on the path in the evolutionary tree from y to z receive a penalty. The edge in the evolutionary tree penalized the least is then the insertion point for s. Ties are broken arbitrarily. This randomized procedure is repeated several times to produce a collection of evolutionary trees $T_1, T_2, \dots, T_k$. From this collection a maximum consensus evolutionary tree T is obtained (Margush and McMorris 1981). The maximum consensus evolutionary tree contains those edges that occur in more than half of the evolutionary trees $T_1, T_2, \dots, T_k$.

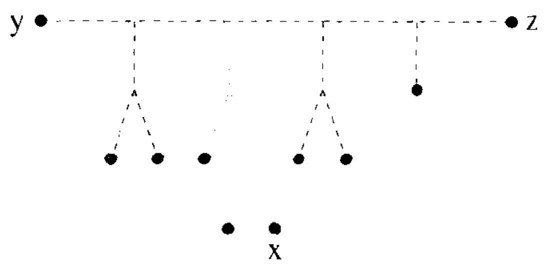


Figure 5.3
Dashed lines are edges that s should not be inserted on if $sx \mid yz \in Q$.

Method Assessment Although the optimization criterion is not explicit, implicitly it attempts to optimize the MQC criterion. The procedure is randomized and repeated in order to avoid local optima in the evolutionary tree space. The probability of selecting a sequence insertion order that yields an optimal or near optimal evolutionary tree, in the sense of the MQC criterion, stands as the most important unresolved question surrounding quartet puzzling. This is crucial because if this probability is small or zero, then it is unlikely or impossible that quartet puzzling will find the optimal tree even with substantial computational resources. In practice, one would simply iterate as many times as computationally feasible.

The motivation for the penalty rule is that an edge should be penalized if the insertion of s onto that edge results in $sx \mid yz$ not being realized by the evolutionary tree. However, the quartet puzzling penalty rule is slightly flawed in that it does not penalize edges in subtrees of the path from y to z excepting the subtree containing x as depicted in figure 5.3.

The maximum consensus evolutionary tree is well-defined in the sense that if two edges appear in more than half of the evolutionary trees $T_1, T_2, \ldots, T_k$, then at least one of these evolutionary trees contains both e_1 and e_2. This implies that e_1 and e_2 are *compatible*, as required. A collection of bipartitions of the sequence set S (i.e., edges) are compatible if they can coexist within the same evolutionary tree. It is well-known that a set of bipartitions is compatible if they are pairwise compatible.

The weakness of the maximum consensus evolutionary tree T is that it may not be resolved despite there being edges in $T_1, T_2, \ldots, T_k$ that could fully resolve T yet do not appear in over half of the evolutionary trees T_i. Hence, quartet puzzling often produces a conservative estimate of the unknown evolutionary tree T_{true}. Various heuristics can be utilized to further resolve this conservative estimate.

Quartet puzzling utilizes unweighted quartet topologies but could easily be extended to utilize both intraweighted and interweighted quartet topologies. Quartet puzzling is publicly available at http://www.tree-puzzle.de/.

5.4.2 Maximum Quartet Consistency

Here we discuss the complexity status of the recombination stage of the quartet method and describe an approximation algorithm that addresses the recombination problem. As discussed above in section 5.3, the recombination stage is typically formulated as the Maximum Quartet Consistency (MQC) problem. Analysis of the complexity status of a problem is important because it informs the algorithm design strategy for solving the problem.

An interesting observation is that the distinction between a complete and incomplete quartet topology set is crucial to the complexity status of the MQC problem. The incomplete version of MQC is NP-hard (Steel 1992) and can easily be shown to be MAX-SNP-hard (Papadimitriou 1994). The complete version of MQC is also NP-hard (Berry et al. 1999); however, it can be approximated with arbitrary accuracy with the approximation algorithm described below (Jiang et al. 1998). This result is fundamental to establishing the feasibility of the quartet method paradigm as it establishes a performance guarantee for quartet topology recombination.

Instances of complete MQC are *dense* relative to instances of incomplete MQC. In recent years, it has been discovered that dense versions of MAX-SNP problems such as Max-Cut, Betweenness, and Max-k-Sat have yielded polynomial time approximation schemes (PTAS) for these problems (Arora et al. 1996, 1995). Dense instances of problems such as Max-Cut are graphs with $\Omega(n^2)$ edges, whereas dense instances of Max-k-Sat are boolean k-Sat formulae with $\Omega(n^k)$ clauses.

Method Overview Let Q be a complete instance of MQC with label set S and let T_{OPT} be an optimal solution. Because a randomly selected tree has a one-third chance of inducing $ab|cd \in Q$, for each quartet $\{a, b, c, d\}$, $|Q_{T_{OPT}} \cap Q| \geq \binom{n}{4}/3$ (Ben-Dor et al. 1998b; Berry 1998). The goal is then to find an approximation algorithm such that

$$|Q_{T_{APX}} \cap Q| \geq |Q_{T_{OPT}} \cap Q| - \epsilon n^4$$

where T_{APX} is the result of the approximation algorithm. The approximation algorithm that accomplishes this is founded upon two concepts: a k-bin decomposition of T_{OPT} and smooth integer polynomial programs.

DEFINITION 2 T_k is a k-bin decomposition of T_{OPT} if there is a partition of S into *bins* $S_1, S_2, \ldots S_k$ such that

- For each S_i, $|S_i| \leq 6n/k$. Furthermore, there is a vertex v_i of degree $|S_i| + 1$, called the *bin root*, that is adjacent to each vertex in S_i.

• For all quartets $\{a, b, c, d\}$ where a, b, c, and d are in different bins of T_k, $ab \mid cd \in Q_{T_{OPT}}$ if and only if $ab \mid cd \in Q_{T_k}$.

There is a k-bin decomposition T_k of T_{OPT} such that $|Q_{T_k} \cap Q| \geq |Q_{T_{OPT}} \cap Q| - (c'/k)n^4$, for some constant c'. To approximate T_{OPT}, it suffices to approximate T_k.

Consider a fixed k and let K be T_k with all leaves removed (and thus the leaves of K are the bin roots of T_k). K is called the *kernel* of T_k, and T_k is called a *completion* of K. K is completed to T_k by providing a label-to-bin assignment.

If the kernel K of T_k is known, then to approximate T_k, it suffices to determine an approximately optimal label-to-bin assignment for K. This problem is formalized as follows:

LABEL-TO-BIN ASSIGNMENT (LBA)
Instance: Set Q of quartet topologies and (degree-3) kernel K with k leaves.
Goal: Find a completion T' of K that maximizes $|Q_{T'} \cap Q|$.

LBA is NP-hard (Bryant 1997) but LBA can be formulated as a smooth integer polynomial problem and a resulting PTAS for LBA defined (Jiang et al. 1998). This PTAS utilizes the fact that the problem instance is dense. In particular, it is shown that for any $\epsilon > 0$, $|Q_{T'} \cap Q| \geq |Q_{\hat{T}} \cap Q| - \epsilon n^4$, where Q and K denote the instance of LBA, T' is the completion of K produced by the PTAS and $\hat{T}$ is an optimal completion of K.

Because k is a constant, for every tree with k leaves, an instance of LBA can be solved approximately in polynomial time. Let T_{APX} be the completed tree obtained that maximizes $|Q_{T_{APX}} \cap Q|$. Because the kernel K of T_k is one of the trees completed, it follows that $|Q_{T_{APX}} \cap Q| \geq |Q_{T'} \cap Q|$ where T' is the completion of K.

Method Assessment It should be noted that although the above PTAS produces an evolutionary tree with an accuracy guarantee it is not yet a practical algorithm. Further research is required in order for it to become efficient enough to solve moderate to large instances. However, the existence of the PTAS suggests that MQC can be efficiently and approximately solved with further research.

5.4.3 Semi-Definite Programming

The semi-definite programming (SDP) approach taken by Ben-Dor, Chor, Graur, Ophir, and Pelleg (Ben-Dor et al. 1998a) is the only quartet method presented here that utilizes a geometric interpretation of the quartet recombination problem.

Method Overview The SDP approach begins with a set Q of quartet topologies where each quartet topology q has an assigned confidence value $c(q)$ obtained using

the bootstrap technique (Felsenstein 1985) although the confidence value could be obtained using other methods. The set Q may be incomplete. The score of an evolutionary tree T, $score(T)$, is defined to be

$$\sum_{q \in Q_T \cap Q} c(q) + \frac{1}{3} \sum_{q \text{ unresolved in } T} c(q)$$

The latter term in the above expression is the expected increase in the score of a random expansion of T if T is not resolved.

The SDP approach then attempts to embed each sequence on the unit sphere in $\mathbb{R}^n$ such that for each quartet topology $ab \mid cd$, the pairs (a, b) and (c, d) are close, whereas the pairs $(a, c), (a, d), (b, c)$, and (b, d) are more distant. This is accomplished by formulating the embedding problem as a semidefinite program with an appropriate objective function. Once an embedding is obtained, a hierarchical clustering algorithm is applied using Euclidean distance to group sequences into a tree.

Method Assessment The SDP approach is a heuristic quartet method that essentially transforms quartet topology information to sequence distance information. It is unclear how faithfully the sphere embedding of sequences represents the quartet topology information or on what basis this distance information is a priori advantageous over other forms of sequence distance information.

Examples of other distance measures that the SDP approach can be compared to include the following:

- For each pair of sequences x and y, define $s(x, y)$ to be the number of quartets of the form $ax \mid by$ (Barthélemy and Guénoche 1991; see also Sattath and Tversky 1977).

  The motivation for $s(x, y)$ is that it is correlated to distance in an evolutionary tree T given that Q approximates Q_T. Hence, $s(x, y)$ relates quartet topology directly to evolutionary similarity.
- For each pair of aligned sequences x and y, define $d(x, y)$ to be

$$-c\frac{3}{4}\ln(1 - \frac{4}{3}D(x, y))$$

where $D(x, y)$ is the Hamming distance between x and y.

The distance $d(x, y)$ is the corrected Jukes-Cantor distance, based on the Jukes-Cantor model of sequence evolution (Jukes and Cantor 1969). Although this distance does not utilize quartet topology, it does use a correlation between sequence similarity and evolutionary tree topology.

In contrast, the SDP approach does not have an evolutionary basis, and so further research is required. Nevertheless, the SDP approach successfully illustrates how advances in another branch of computer science (solving semidefinite programs efficiently) can be leveraged to solve quartet recombination.

It should be noted that Ben-Dor et al. (1998a) present an efficient (but still exponential) algorithm for solving quartet recombination exactly. This was later improved upon by Bryant and Steel (1999).

5.4.4 Short Quartet Method

The key idea behind the short quartet method, introduced by Erdös, Rice, Steel, Székely, and Warnow (Erdös et al. 1997), is that there is a subset of quartet topologies in Q_T that is sufficient to recover the evolutionary tree T and that tend to be more accurately inferred. The short quartet method identifies these quartet topologies and utilizes them to reconstruct an evolutionary tree.

Method Overview Let e be an edge in the evolutionary tree T and let $\{a, b, c, d\}$ be a quartet such that a, b, c, and d are sequences in the four subtrees of T induced by removing e and its endpoints from T. The quartet $\{a, b, c, d\}$ is called a *short quartet* if $max(\{a, b, c, d\})$ is minimum among all such quartets where

$$max(\{a, b, c, d\}) = max(d_T(a, b), d_T(a, c), d_T(a, d), d_T(b, c), d_T(b, d), d_T(c, d))$$

T can be reconstructed efficiently from its set of short quartet topologies (Erdös et al. 1997).

Let D be a distance matrix obtained from the sequence data. The *weak four-point method* can be used to infer a set Q of quartet topologies from D:

$$ab\,|\,cd \in Q \Leftrightarrow D(a, b) + D(c, d) < min(D(a, c) + D(b, d), D(a, d) + D(b, c))$$

The weak four-point method is a variation on the four-point method motivated by the fact that D is an approximation to an evolutionary tree metric, and so, equality between $D(a, c) + D(b, d)$ and $D(a, d) + D(b, c)$ is unlikely to be observed.

For a threshold t define Q_t to be those quartet topologies $ab\,|\,cd \in Q$ such that $max(\{a, b, c, d\}) \leq t$. If t is sufficiently large and all short quartet topologies in Q are correctly inferred, then Q_t will contain all short quartets of the unknown evolutionary tree T_{true}. T_{true} can then be reconstructed from Q_t (Erdös et al. 1997). The details of the reconstruction are omitted.

Method Assessment The motivation for the short quartet method is that short quartet topologies are sufficient for recovering the unknown evolutionary tree and that

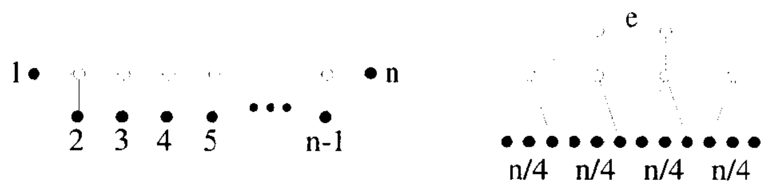


Figure 5.4
Effects of topology on the short quartet method.

they are more accurately inferred than longer quartet topologies, for example, because long branch attraction can be avoided (Hendy and Penny 1982). This assumption can be violated in several ways. First, depending on the topology of the evolutionary tree, short quartet topologies do not always avoid long branch attraction. Second, very short branch lengths can also be problematic, and so very short quartet topologies are not necessarily more accurate. Third, the short quartet method is not robust to error—it will fail even if only one of the short quartet topologies is erroneous.

To illustrate the dependence of the short quartet method on the topology of the underlying evolutionary tree, consider the two evolutionary trees in figure 5.4. The leftmost evolutionary tree is called a caterpillar due to its linear structure. Here the number of short quartets is $n - 1$. For the rightmost evolutionary tree, consisting of four subtrees each with $n/4$ leaves, the number of short quartets across the edge e alone is $n^4/256$. Clearly the ratio of short quartets to total number of quartets $\binom{n}{4}$ varies greatly with topology. Specifically, the caterpillar requires as mall specific set of quartet topologies to be inferred correctly, whereas the rightmost evolutionary tree requires most of its quartet topologies to be inferred correctly. A thorough examination of the short quartet method requires that the method be assessed on a wide range of evolutionary tree topologies.

Despite these concerns, the short quartet method introduces an interesting and important observation: some quartet topologies are more important than others for recovering an evolutionary tree. Related work examines the question of which incomplete subsets of Q_T can be extended uniquely to obtain T (Bryant and Steel 1995). For example, an incomplete set of quartet topologies can be extended using inference rules such as the following (these can be easily verified by examining all evolutionary trees consistent with the given assumptions):

- If $ab \mid cd$ and $ab \mid ce \in Q_T$ then $ab \mid de \in Q_T$.
- If $ab \mid cd$ and $ac \mid de \in Q_T$ then $ab \mid ce$, $ab \mid de$ and $bc \mid de \in Q_T$.
- If $ab \mid cd$, $ab \mid ef$ and $ce \mid df \in Q_T$ then $ab \mid df \in Q_T$.

It is known that there are inference rules of arbitrarily high order (Bryant and Steel 1995).

The short quartet method utilizes the fact that the application of the first two inference rules above to a set of short quartet topologies will yield a complete set of quartet topologies.

Unlike the other quartet methods presented here, the short quartet method is not heuristic, nor does it solve an optimization problem. Rather, it specifies the requirements needed in order to recover the unknown evolutionary tree exactly and then determines whether these requirements are met by the data.

Finally, the short quartet method is a fast converging method. This means that the short quartet method requires relatively short sequences to converge upon the correct evolutionary tree. This is an important property because, in practice, the amount of sequence data available is limited. To establish fast convergence one must assume a model of evolution, and so fast convergence results hold as long as these assumptions are not violated.

5.4.5 Hypercleaning

The topology of an evolutionary tree can be specified by its set of edge-induced bipartitions. One approach to recovering an evolutionary tree is to first recover all bipartitions highly supported by the sequence data and then to select from these bipartitions a compatible set (see section 5.4.1) of bipartitions. To accomplish this, the following must be defined:

- a bipartition support function,
- a method for obtaining highly supported bipartitions, and
- a method for selecting a subset of compatible bipartitions.

This approach is illustrated by hypercleaning (Berry et al. 2000).

Bipartition Support Let Q be a set of inferred quartet topologies. For a given bipartition (X, Y), define $Q_{(X,Y)}$ to be the set of quartet topologies of the form $xx' \mid yy'$ where $x, x' \in X$, and $y, y' \in Y$. $Q_{(X,Y)}$ is the set of quartet topologies induced by the bipartition (X, Y). Bipartition support is defined in terms of the amount of disagreement between Q and $Q_{(X,Y)}$. Define $ab \mid cd$ to be a quartet error *across* bipartition (X, Y) if $ab \mid cd \in Q_{(X,Y)} - Q$. The *normalized distance* from a set of quartets Q to a bipartition (X, Y) is defined to be

$$\delta(Q, (X, Y)) = \frac{4|Q_{(X,Y)} - Q|}{|X|(|X| - 1)|Y|(|Y| - 1)}$$

where the number of quartet topologies in $Q_{(X,Y)}$ is $|X|(|X|-1)|Y|(|Y|-1)/4$. Normalization permits the comparison of the support values for two different bipartitions. When (X, Y) is *trivial* ($|X| = 1$ or $|Y| = 1$), the normalized distance is defined to be 0.

Recovering Neighborhoods of Q The task of recovering bipartitions highly supported by Q is the task of generating a bipartition neighborhood of Q

$$\{(X, Y) \mid \delta(Q, (X, Y)) \leq r\}$$

which is called the closed r-neighborhood of Q. When the inequality is strict, it is called the open r-neighborhood of Q.

The closed 0-neighborhood of Q corresponds to those bipartitions that have 0 quartet topology differences with Q. There is an $O(n^4)$ time algorithm, called the Q^* method, for recovering this set of bipartitions (Berry and Gascuel 1997). However, the Q^* tree, also known as the Buneman tree (Buneman 1971), is a very conservative estimate of T as it includes only those bipartitions with 0 quartet topology difference with Q.

The open $\frac{2}{|X||Y|}$-neighborhood of Q is known to be compatible (Berry et al. 1999) and there is an $O(n^5)$ algorithm for recovering this neighborhood from Q (Berry et al. 2000). Note that the closed $\frac{2}{|X||Y|}$-neighborhood of Q is not necessarily compatible (Jiang et al. 1998).

Although the open $\frac{2}{|X||Y|}$-neighborhood of Q is compatible, it is not likely to return all $n-3$ compatible, nontrivial bipartitions of the underlying evolutionary tree T_{true} (Berry et al. 2000). In order to include more edges of T_{true}, a parameter $m > 0$ is introduced and the following neighborhood of Q is defined

$$Best(Q, m) = \left\{(X, Y) \mid \delta(Q, (X, Y)) < \frac{2m}{|X||Y|}\right\}$$

Thus the set $Best(Q, m')$ contains the set $Best(Q, m)$ for all $m' \geq m$. Increasing the value of m increases the neighborhood of Q, including bipartitions more weakly supported by Q.

$Best(Q, m)$ can be obtained by hypercleaning in time polynomial in n but exponential in m (Berry et al. 2000). Specifically, a time bound on hypercleaning is $O(n^5 f(2m) + n^7 f(m))$ time where $f(m) = 4m^2(1+2m)^{4m}$. The hypercleaning algorithm has many properties that make it useful in practice:

- Although the running time of hypercleaning is exponential in m, in practice, only small values of m (≤ 5) are required in order to recover all edges of T_{true}. In fact, hypercleaning can be successfully applied to relatively large sequence data sets (one hundred's of sequences) (Badger et al. 2000).
- The running time of hypercleaning varies with m, which is exponential in the amount of quartet error in the bipartitions being recovered. Hence, highly supported edges of T_{true} are recovered more quickly than poorly supported edges. In practice, hypercleaning can be applied to very large sequence data sets to obtain all highly supported edges of T_{true}. The sequence data set can then be subdivided into smaller subproblems using the edges recovered.
- Hypercleaning is an exact algorithm and not a heuristic. That is, it satisfies the following accuracy guarantee:

THEOREM 4 The hypercleaning algorithm recovers all bipartitions (X, Y) in the underlying evolutionary tree T_{true} with fewer than $m(|X|-1)(|Y|-1)/2$ quartet errors.

If Q is reasonably correlated to Q_T, then hypercleaning is a powerful tool for estimating evolutionary trees.

- Hypercleaning finds all highly supported alternative bipartitions, which gives the researcher a sense of the uniqueness of the evolutionary tree produced.

Selecting Bipartitions from $Best(Q,m)$ $Best(Q,m)$ may be very large, and so will contain incompatible bipartitions. A greedy algorithm can be defined to select a compatible subset of $Best(Q,m)$. Let $(X_1, Y_1), (X_2, Y_2), \ldots, (X_k, Y_k)$ be the bipartitions in $Best(Q,m)$ ordered by increasing normalized distance to Q. The greedy algorithm selects the following subset, called $Comp(Q,m)$, of $Best(Q,m)$:

- $(X_1, Y_1) \in Comp(Q,m)$
- $(X_j, Y_j) \in Comp(Q,m)$ if (X_j, Y_j) is compatible with all $(X_i, Y_i) \in Comp(Q,m)$ where $i < j$.

Observe that $Comp(Q,m)$ is a set of compatible bipartitions and can be easily obtained from $Best(Q,m)$.

Note that the above simple algorithm is only a heuristic for selecting the maximal set of compatible bipartitions that minimizes the sum of normalized distances to Q. For other criteria, there are exact polynomial time algorithms. For example, having inferred a set $Best(Q,m)$ with enough edges to construct an evolutionary tree, a maximal set of compatible bipartitions minimizing

$max_{(X,Y)}\{\delta(Q,(X,Y))\}$

which is the L_∞ norm on bipartitions can be obtained using the exact polynomial time algorithm (Bryant 1997), which has time complexity $O(|Best(Q, m)|^2)$.

Further research into the efficient and accurate selection of compatible bipartitions from $Best(Q, m)$ is required.

5.5 Closing Remarks and Resources

The development of computational methods for the evolutionary analysis of gene sequences is simultaneously complicated and enriched by the need for biological relevance and computational feasibility. Although much progress has been made, it is unclear if the research has resulted in more answers than new questions. Some of the major challenges facing phylogenetics include:

- The development of computational methods for inferring large evolutionary trees. Current datasets far exceed the capacity of current accurate computational methods. As sequencing technology improves and becomes more automated, this gap will widen.
- The development of integrated, interactive, and graphical methods for evolutionary analyses. Due to the large and complex datasets now available, computational methods that can incorporate diverse evolutionary data are required. Furthermore, methods need to be more interactive and graphical in order for the scientist to conceptualize the entire evolutionary history of large datasets.
- The development of benchmarks and techniques for comparing phylogenetic techniques. The diversity of phylogenetic methods, even when considering only quartet methods, makes comparison of methods challenging. It is unclear under which conditions a given phylogenetic method will outperform another method.

Some phylogenetic resources:

- Phylip: http://evolution.genetics.washington.edu/phylip.html
A collection of phylogenetic tools maintained by Joe Felsenstein.
- Tree of Life: http://phylogeny.arizona.edu/tree/phylogeny.html
The Tree of Life is a project containing information about the diversity of organisms on Earth, their history, and characteristics. Presented in an easy to navigate format. Maintained by David Maddison.
- Green Plant Phylogeny: http://ucjeps.berkeley.edu/bryolab/greenplantpage.html
A repository of green plant evolutionary information and large data sets.

• Ribosomal RNA Database Project: http://www.cme.msu.edu/RDP/html/index.html
A repository of ribosomal RNA evolutionary information including large data sets that are easily accessed and navigated.

• TreeBASE http://herbaria.harvard.edu/treebase/
TreeBASE stores phylogenetic trees and the data matrices used to generate them from published research papers.

• Phylogenetic Resources http://www.ucmp.berkeley.edu/subway/phylogen.html
Information including software, meetings, databases, publications, and societies of interest to evolutionary biologists.

Acknowledgments

The author would like to acknowledge the support of CITO and NSERC, as well as the anonymous referees for suggested improvements to the chapter.

References

Arora, S., Frieze, A., and Kaplan, H. (1996). A new rounding procedure for the assignment problem with applications to dense graph arrangement problems. In *37th Annual Symposium on Foundations of Computer Science*, 21–30.

Arora, S., Karger, D., and Karpinski, M. (1995). Polynomial time approximation schemes for dense instances of *np*-hard problems. In *Proceedings of the Twenty-Seventh Annual ACM Symposium on the Theory of Computing*, 284–293. New York: ACM Press.

Badger, J., Hu, M., and Kearney, P. (2000). Inferring large evolutionary trees. *Manuscript*.

Badger, J., and Kearney, P. (2000). Picking fruit from the tree of life: Comments on taxonomic sampling and the quartet method. In *Proceedings of the 16th ACM Symposium on Applied Computing 2001 (SAC2001)*. New York: ACM Press.

Barker, W. C., and Dayhoff, M. O. (1980). Evolutionary and functional relationships of homologous physiological mechanisms. *BioScience* 30: 593–600.

Barthélemy, J.-P., and Guénoche, A. (1991). *Trees and Proximity Representations*. New York: Wiley.

Ben-Dor, A., Chor, B., Graur, D., Ophir, R., and Pelleg, D. (1998a). Constructing phylogenies from quartets: Elucidation of eutherian superordinal relationships. *Journal of Computational Biology* 5(3): 377–390.

Ben-Dor, A., Chor, B., Graur, D., Ophir, R., and Pelleg, D. (1998b). From four-taxon trees to phylogenies: The case of mammalian evolution. In *Proceedings of the Second Annual International Conference on Computational Molecular Biology*, 9–19. New York: ACM Press.

Benner, S. A., Cannarozzi, G., Chelvanayagam, G., and Turcotte, M. (1997). *Bona fide* predictions of protein secondary structure using trans-parent analyses of multiple sequence alignments. *Chem. Rev.* 97: 2725–2843.

Benner, S. A., Cohen, M. A., and Gonnet, H. H. (1993). Empirical and structural models for insertions and deletions in the divergent evolution of proteins. *J. Mol. Biol.* 229: 1065–1082.

Berry, V. (1998). Personal communications.

Berry, V., Bryant, D., Jiang, T., Kearney, P., Li, M., Wareham, T., and Zhang, H. (2000). A practical algorithm for recovering the best supported edges of an evolutionary tree. In *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms*, 287–296.

Berry, V., and Gascuel, O. (1997). Inferring evolutionary trees with strong combinatorial evidence. In *Proceedings of the Third Annual International Computing and Combinatorics Conference* (COCOON), 111–123. LNCS 1276, Berlin: Springer.

Berry, V., Jiang, T., Kearney, P., Li, M., and Wareham, T. (1999). Quartet cleaning: Improved algorithms and simulations. In *European Symposium on Algorithms* (ESA), 313–324. Berlin: Springer.

Bryant, D. (1997). *Structures in Biological Classification.* Ph.D. dissertation, Department of Mathematics and Statistics, University of Canterbury.

Bryant, D., and Steel, M. (1995). Extension operations on sets of leaf-labelled trees. *Advances in Applied Mathematics* 16: 425–453.

Bryant, D., and Steel, M. (1999). Fast algorithms for constructing optimal trees from quartets. In *Proceedings of the 10th Annual ACM-SIAM Symposium on Discete Algorithms*, 147–155. New York: ACM Press.

Buneman, P. (1971). The recovery of trees from measures of dissimilarity. In *Mathematics in the Archaeological and Historical Sciences*, Hodson, F., Kendall, D., and Tautu, P., eds., 387–395. Edinburgh: Edinburgh University Press.

Colonius, H., and Schulze, H. (1981). Tree structures for proximity data. *British Journal of Mathematical and Statistical Psychology* 34: 167–180.

Doolittle, R. F. (1986). *Of Urfs and Orfs: A Primer on How to Analyze Derived Amino Acid Sequences.* Mill Valley, Calif.: University Science Books.

Erdös, P., Steel, M., Székely, L., and Warnow, T. (1997). Constructing big trees from short sequences. In *Proceedings of the 24th International Colloquium on Automata, Languages, and Programming* (ICALP), Berlin: Springer.

Farris, J. S. (1970). Methods for computing Wagner trees. *Sys. Zool.* 34: 21–34.

Felsenstein, J. (1978a). Cases in which parsimony and compatibility will be positively misleading. *Sys. Zool.* 27: 401–410.

Felsenstein, J. (1978b). Cases in which parsimony and compatibility will be positively misleading. *Sys. Zool.* 27: 401–410.

Felsenstein, J. (1981). Evolutionary trees from DNA sequences: A maximum likelihood approach. *J. Mol. Evol.* 17: 368–376.

Felsenstein, J. (1985). Confidence limits on phylogenies: An approach using the bootstrap. *Evolution* 39: 164–166.

Felsenstein, J. (1988). Phylogenies from molecular sequences: Inference and reliability. *Ann. Rev. Genet.* 22: 521–565.

Feng, D., and Doolittle, R. F. (1987). Progressive sequence alignment as a prerequisite to correct phylogenentic trees. *J. Mol. Evol.* 25: 351–360.

Graybeal, A. (1998). Is it better to add taxa or characters to a difficult phylogenetic problem? *Sys. Biol.* 47: 9–17.

Hakimi, S. L., and Yau, S. S. (1964). Distance matrix of a graph and its realizability. *Q. Appl. Math.* 22: 305–317.

Hendy, M. D., and Penny, D. (1982). Branch and bound algorithms to determine minimal evolutionary trees. *Math. Biosci.* 59: 277–290.

Hendy, M. D., and Penny, D. (1989). A framework for the quantitative study of evolutionary trees. *Sys. Zool.* 38: 297–309.

Hillis, D. M. (1996). Inferring complex phylogenies. *Nature* 383: 130.

Huelsenbeck, J., and Hillis, D. (1993). Success of phylogenetic methods in the four-taxon case. *Sys. Biol.* 42(3): 247–264.

Jiang, T., Kearney, P. E., and Li, M. (1998). Orchestrating quartets: Approximation and data correction. In *Proceedings of the 39th IEEE Symposium on Foundations of Computer Science*. IEEE Computer Society, Los Alamitos, CA 416–425.

Jukes, T. H., and Cantor, C. R. (1969). Evolution of protein molecules. In *Mammalian Protein Metabolism*, Munro, H. N., ed., 21–123. New York: Academic Press.

Kim, J. (1996). General inconsistency conditions for maximum parsimony: Effects of branch lengths and increasing numbers of taxa. *Sys. Biol.* 45: 363–374.

Lecointre, G., Philippe, H., Le, H., and Le Guyader. H. (1993). Species sampling has a major impact on phylogenetic inference. *Mol. Phyl. Evol.* 2: 205–224.

Li, M., Badger, J., Chen, X., Kwong, K., Kearney, P., and Zhang, H. (2001). An information based sequence distance and its application to whole genome mitochondrial phylogeny. *Bioinformatics* 17: 149 154.

Li, W.-H. (1997). *Molecular Evolution*. Sunderland, Mass.: Sinauer Associates, Inc.

Losos, J. B., and Adler, F. R. (1995). Stumped by trees? A generalized null model for patterns of organismal diversity. *Am. Naturalist* 145(3): 329–342.

Maidak, B. L., Cole, J. R., Parker, C. T., Jr, G. M. G., Larsen, N., Li, B., Lilburn, T. G., McCaughey, M. J., Olsen, G. J., Overbeek, R., Pramanik, S., Schmidt, T. M., Tiedje, J. M., and Woese, C. R. (1999). A new version of the RDP (Ribosomal Database Project). *Nucl. Acids Res.* 27: 171 173.

Margoliash, E. (1963). Primary structure and evolution of cytochrome c. *Proc. Natl. Acad. Sci. USA* 50: 672–679.

Margush, T., and McMorris, F. (1981). Consensus n-trees. *Bull. Math. Biol.* 43: 239 244.

Olsen, G. J., Matsuda, H., R., H., and Overbeek, R. (1994). Fastdnaml: A tool for construction of phylogenetic trees of DNA sequences using maximum likelihood. *CABIOS* 10: 41 48.

Papadimitriou, C. (1994). *Computational Complexity*. New York: Addison-Wesley.

Poe, S. (1998). Sensitivity of phylogeny estimation to taxonomic sampling. *Sys. Biol.* 47: 18 31.

Saitou, N., and Nei, M. (1987). The neighbor-joining method: A new method for reconstructing phylogenetic trees. *Mol. Biol. Evol.* 4: 406–425.

Sattath, S., and Tversky, A. (1977). Additive similarity trees. *Psychometrika* 42: 319 345.

Slowinski, J. B., and Guyer, C. (1989). Testing the stochasticity of patterns of organismal diversity: An improved null model. *Am. Naturalist* 134(6): 907–921.

Smith, K., and Warnow, T. (1998). Taxon sampling and accuracy of evolutionary tree reconstruction. In *DIMACS Symposium on Large Phylogenetic Tree Reconstruction*, DIMACS, Rutgers University, NJ.

Steel, M. (1992). The complexity of reconstructing trees from qualitative characters and subtrees. *Journal of Classification* 9: 91–116.

Strimmer, K., and von Haeseler, A. (1996). Quartet puzzling: A quartet maximum-likelihood method for reconstructing tree topologies. *Mol. Biol. Evol.* 13(7): 964–969.

Swofford, D. L., Olsen, G. J., Waddell, P. J., and Hillis, D. M. (1996). Phylogenetic inference. In *Molecular Systematics, 2nd ed.*, Hillis, D. M., Moritz, C., and Mable, B. K., eds., 407 514. Sunderland Mass.: Sinauer Associates.

Waterman, M. S., Smith, T. F., Singh, M., and Beyer, W. A. (1977). Additive evolutionary trees. *J. Theoret. Biol.* 64: 199–213.

6 Genome Rearrangement

David Sankoff and Nadia El-Mabrouk

6.1 *Introduction*

The difference between genome rearrangement theory and other approaches to comparative genomics, and indeed most other topics in computational molecular biology, is that it is not directly based on macromolecular sequences, either nucleic acids or proteins. Rather like classical genetics: its building blocks are genes, and the structures of interest are chromosomes, abstracted in terms of the linear order of the genes they contain. Of course, genes and their RNA and protein products are macromolecules, but here we do not focus on the internal structure of genes and assume that the problems of determining the identity of each gene, and its homologs in other genomes, have been solved, so that a gene is simply labeled by a symbol indicating the class of orthologs to which it belongs. Moreover, the linearity of chromosomal structure does not evolve by a nucleotide substitution process in the way DNA does, or even by the same type of insertion/deletion processes, but by a number of very different rearrangement processes that are nonlocal, the scope of which may involve an arbitrarily large proportion of a chromosome. As a consequence, the formal analysis of rearrangements bears little resemblance in detail to DNA or protein comparison algorithms.

Nevertheless, in analogy with sequence comparison, the study of genome rearrangements has focused on inferring the most economical explanation for observed differences in gene orders in two or more species, as represented by their genomes, in terms of a small number of elementary processes. After first formalizing in section 6.2 the notion of a genome as a set of chromosomes, each consisting of an ordered set of genes, we will proceed in section 6.3 to a survey of genomic distance problems. More detail on the Hannenhalli-Pevzner theory for "signed" distances follows in section 6.4. Section 6.5 will be devoted to phylogenetic extensions, and section 6.6 to problems of gene and genome duplication and their implications for genomic distance and genome-based phylogeny.

6.2 The Formal Representation of the Genome

As a first approximation, a genome can be thought of as a set containing on the order of 10^3 (some bacteria) to 10^5 (human) distinct elements called genes. In more realistic analyses, it may be necessary to consider that some genes occur with a multiplicity of

two or higher in a genome, which cannot be captured in a set formulation. This situation will be explored in section 6.6.

6.2.1 Synteny

The genes in plants, animals, yeasts, and other eukaryotes are partitioned among a number of chromosomes, generally between 10 and 100, though it can be as low as two or three (Jackson 1957; Lima-de Faria 1980), or much higher than 100. Two genes located on the same chromosome in a genome are said to be *syntenic* in that genome.

Some genome rearrangements involve parts of one chromosome being relocated to another chromosome. Syntenic structure is generally different between different species and usually identical among all the members of a single species. A few species tolerate population heterogeneity involving small differences in syntenic structure, where heterokaryotypic individuals are not only viable, but fertile (McAllister 2000).

In prokaryotic genomes, comprising both eubacteria and archaebacteria, the genome typically resides on a single chromosome. Organelles, such as the mitochondria found in most eukaryotes and the chloroplasts in plants and algae, also have relatively small single-chromosome genomes, containing fewer than a hundred (mitochondria) or 250 (chloroplasts) genes, and are believed to be the highly reduced descendants of prokaryotic endosymbionts.

6.2.2 Order and Polarity

Syntenic structure, as we shall see in section 6.3.6, suffices to initiate the study of genome rearrangements. Two additional levels of chromosomal structure, when they are available, add valuable information about rearrangement. The first is gene order. The genes on each chromosome have a linear order that is characteristic of each genome. Note that although our discussion in this paper is phrased in terms of the order of genes along a chromosome, the key aspect for mathematical purposes is the order and not the fact that the entities in the order are genes. They could as well be blocks of genes contiguous in the two (or N) species being compared, conserved chromosomal segments in comparative genetic maps (cf. Nadeau and Sankoff 1998) or, indeed, the results of any decomposition of the chromosome into disjoint ordered fragments, each identifiable in the two (or in all N) genomes.

The next level of structure is the transcription direction associated with each gene. In the double-stranded DNA of a genome, typically some genes are found on one strand and are read in the direction associated with that strand, whereas other genes are on the complementary strand that is read in the opposite direction. To capture

this distinction in the mathematical notation for a genome, the genes on one strand are designated as of positive polarity and those on the other as negative. The latter are written with a minus sign preceding the gene label. Genome distance problems where this level of structure is known and taken into account are called "signed," in contrast to the "unsigned" case, where no directional information is known (or used).

6.2.3 Linearity versus Circularity

In eukaryotes such as yeast, amoeba, or humans, the genes on a chromosome are ordered linearly. There is no natural left-to-right order, no structural asymmetry or polarity between one end of a chromosome and the other. Biologists distinguish between the short and long "arms" of a chromosome for nomenclatural purposes, and although we shall see in section 6.2.4 that this has a structural basis, there is no biological reason to order the long arm before the short arm, or vice-versa.

In prokaryotes and organelles, the single chromosome is generally circular. This leads to terminological and notational adjustments—the arbitrariness of left-to-right order becomes the arbitrariness of clockwise versus counterclockwise ordering, and the notion of one gene appearing in the order somewhere before another is no longer meaningful. Most computational problems in genome comparison are no more difficult for circular genomes than linear ones, though there is one clear exception where the circular problem is much harder, as described in section 6.3.1.

6.2.4 Centromeres and Telomeres

Two structural aspects of eukaryote chromosomes are especially pertinent to genome rearrangements. The centromere is a structurally specialized noncoding region of the DNA, situated somewhere along the length of the chromosome, physically associated with specific proteins. It plays a key role in assuring the proper allocation of chromosomes among the daughter cells during cell division. The centromere divides the chromosome into two arms, both of which normally contain genes. The end of each arm is the telomere, also consisting of noncoding DNA in association with particular proteins.

Because the telomere "protects" the end of the chromosome and is generally necessary in cell division, as is the centromere, genome rearrangements usually do not involve the telomere and do not entail the creation of a chromosome without a centromere or with more than one centromere, though on the evolutionary time scale there are exceptions. New centromeres occasionally emerge remote from existing centromeres and take over the role of the latter, which quickly lose their erstwhile function. Chromosomes sometimes fuse in an end-to-end manner, involving the

loss of two telomeres and a centromere; sometimes the opposite process, fission, also occurs.

6.2.5 Multigene Families

Implicit in the rearrangements literature is that both genomes being compared contain an identical set of genes and the one-to-one homologies (orthologies) between all pairs of corresponding genes in the two genomes have previously been established. Although this hypothesis of *unique genes* may be appropriate for some small genomes, such as viruses and mitochondria, it is clearly unwarranted for divergent species where several copies of the same gene, or several homologous (paralogous) genes—a *multigene family*—may be scattered across a genome.

The Pertinence of Sequence Comparison We stressed at the outset that genome rearrangement analysis is usually carried out separately from, and subsequent to, gene homology assessments. A partial exception to this must be made in the study of multigene families, where we must take into account *degrees* of homology, so that the input data are more subtle than the binary distinction between homologous genes and unrelated genes.

6.3 Operations and Distances

There are many ways of comparing two linear (or circular) orders on a set of objects. In subsection 6.3.1, we first discuss one that is not directly based on any biologically motivated model. In subsection 6.3.2, we introduce a distance that is motivated by general characteristics of genome rearrangements. In the remainder of this section, we review the many edit distances that are based on particular types of rearrangement.

6.3.1 Alignment Traces

One of the earliest suggestions for comparing genomes was to adapt concepts of alignment in sequence comparison, in particular the notion of the trace of an alignment. In its graphic version, this requires displaying the n genes in each of the two genomes, ordered from left to right, one genome above the other, and connecting each of the n pairs of homologous genes with a line. The number of intersections between pairs of lines is a measure of how much one genome is scrambled with respect to the other (Sankoff and Goldstein 1989). (In a classical sequence alignment, there are no intersections.) For linear orders, this measure is easily calculated and analytical tests are available for detecting nonrandom similarities in order; the cir-

cular case is much more difficult. The problem has to do with the optimal alignment of the two genomes, where one circular genome is superimposed on the other and rotated in such a way as to minimize the number of intersections between trace lines connecting genes in the two genomes (Sankoff et al. 1990; Bafna et al. 2000).

6.3.2 Breakpoints

Because genome rearrangements generally involve incorrectly repaired breaks between adjacent genes, it seems appropriate to focus on adjacencies when comparing rearranged genomes. For two genomes X and Y, we define $b(X, Y)$ to be the number of pairs of genes that are adjacent in genome X but not in Y. The easily calculated measure b was first defined in the context of genome rearrangements by Watterson et al. (1982), but was already implicit much earlier in cytogenetic assessments of chromosomal evolution. For signed genomes, the notion of adjacency requires that the configuration of transcription directions be conserved, so that if genome X contains two genes ordered as gh, then these two genes are adjacent in Y only if they occur as gh or as $-h-g$.

The breakpoint distance can be extended to apply to two genomes X and Y that do not contain identical sets of genes. Here we create two smaller genomes X' and Y' by simply deleting those genes that are only in one of the genomes. Then the "induced breakpoint" distance $b_I(X, Y)$ between X and Y is defined to be $b(X', Y')$. For multiple comparisons, as in phylogenetic applications, it is preferable to use the normalized measure $b_N(X, Y) = b_I(X, Y)/l$, where l is the number of genes in X' and Y'.

6.3.3 Edit Distances

Distance problems motivated by particular types of rearrangement processes require calculating an edit distance between two linear or circular orders on the same set of objects, representing the ordering of homologous genes in two genomes. The elementary edit operations may include one or more of the processes depicted in figure 6.1.

6.3.4 Reversal Distances

Reversal, or **inversion**, reverses the order of any number of consecutive terms in the ordered set, which, in the case of signed orders, also changes the sign of each term within the scope of the reversal. Kececioglu and Sankoff (1995) re-introduced the problem—earlier posed by Watterson et al. (1982), and even earlier in the genetics literature, such as in Sturtevant and Novitski (1941)—of computing the minimum reversal distance between two given permutations in the unsigned case, and gave ap-

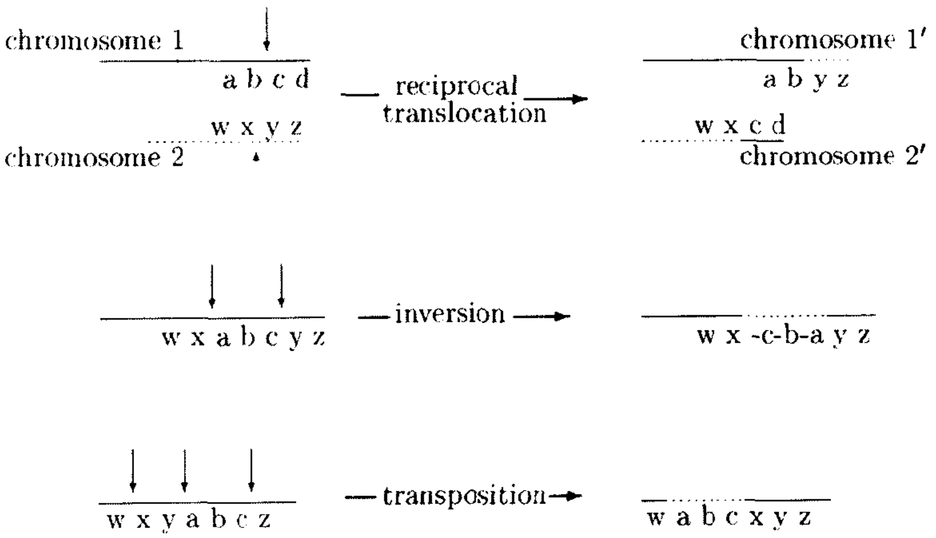


Figure 6.1
Schematic view of genome rearrangement processes. Letters represent positions of genes. Vertical arrow at left indicates breakpoints introduced into original genome. Reciprocal translocation exchanges end segments of two chromosomes. Reversal (or inversion) reverses the order and sign of genes between two breakpoints (dotted segment). Transposition removes a segment defined by two breakpoints and inserts it at another breakpoint (dotted segment) in the same chromosome or another.

proximation algorithms and an exact algorithm feasible for moderately long permutations. Bafna and Pevzner (1996) gave improved approximation algorithms and Caprara (1997) showed this problem to be NP-complete. Recent progress on practical solutions was presented by Caprara et al. (2000). For the signed case, Kececioglu and Sankoff (1994) found tight lower and upper bounds and implemented an exact algorithm that worked rapidly for long permutations. Indeed, Hannenhalli and Pevzner (1999) showed that the signed problem is only of polynomial complexity. Their algorithm was improved by Berman and Hannenhalli (1996) and by Kaplan et al. (2000). We will return to the Hannenhalli-Pevzner approach in sections 6.4 and 6.6.

6.3.5 Transposition Distance

A *transposition* moves any number of consecutive terms from their position in the order to a new position between any other pair of consecutive terms. Computation of the transposition distance between two permutations was considered by Bafna and Pevzner (1998) and Christie (1999), but its NP-completeness has not yet been confirmed. This has been more difficult to analyze than the reversals distance problem (Meidanis and Dias 2000).

6.3.6 Translocation Distance

Kececioglu and Ravi (1995) began the investigation of translocation distances. Hannenhalli (1996) showed that the problem is of polynomial complexity for signed genomes, using methods similar to the reversals distance algorithm.

Syntenic Distance Ferretti et al. (1996) proposed a relaxed form of translocation distance applicable when chromosomal assignment of genes, but not their order, is known. Let A and B be two chromosomes, considered to be sets of genes. A translocation then transforms A and B into $(A - A') \cup B'$ and $(B - B') \cup A'$, respectively, where at least one of A' and B' is a proper subset of A or B. A fusion occurs when, for example, $A' = A$ and $B' =$ the null set, and a fission when either A or B is replaced by the null set, in this formulation.

Then the syntenic distance between two genomes G and H, considered as two different partitions of the same set into subsets (chromosomes), is defined to be the minimum number of translocations necessary to transform G into H. The complexity of its calculation was shown to be NP-complete by DasGupta et al. (1998). Its structure was further investigated by Liben-Nowell (1999) and Kleinberg and Liben-Nowell (2000).

6.3.7 Combined Distances

Distances based on single operations may be of mathematical interest and are appropriate starting points for investigating genomic rearrangements, but realistic models must allow for several types of operation. Several studies have attempted this. The most successful is the extension of the Hannenhalli-Pevzner (1995) theory to cover the case where both translocation and reversal operations are considered.

Another exact polynomial algorithm extending the Hannenhalli-Pevzner theory applies to two genomes that do not have the identical set of genes. This requires calculating the minimum number of reversals and insertions or deletions of contiguous segments of the chromosome necessary to convert one genome into another (El-Mabrouk 2000).

There have also been a number of studies combining transposition and reversals (Gu et al. 1997; Walter et al. 1998), with partial results.

An edit distance that is a weighted combination of inversions, transpositions, and deletions has been studied by Sankoff (1992), Sankoff et al. (1992) and Blanchette et al. (1996). Dalevi et al. (2000) developed a simulation-based method for determining appropriate weighting parameters in the context of prokaryotic evolution. They applied this to the divergence of *Chlamydia trachomatis* and *Chlamydia pneumoniae* (see also Andersson and Eriksson 2000). Their results quantify a propensity

for shorter rather than longer inversions, similar to that reported for eukaryotes by McLysaght et al. (2000).

6.4 The Hannenhalli-Pevzner Theory

In this section, we introduce the structures necessary to understand the results of the three polynomial-time algorithms devised by Hannenhalli and Pevzner. In particular, we sketch how they calculate the edit distance between two genomes, although we do not enter into the details of how they recover the actual operations that convert one of the genomes into the other.

Given two genomes, H_1 and H_2, containing the same genes, where each gene appears exactly once in each genome, the genome rearrangement problem is to find the minimum number of rearrangement operations necessary to transform H_1 into H_2 (or H_2 into H_1). Polynomial algorithms were designed for the reversals-only version of the problem (in the case of single-chromosome genomes) (Hannenhalli and Pevzner 1999), the translocations-only version (Hannenhalli 1996), and the version with both reversals and translocations (Hannenhalli and Pevzner 1995) (the latter two for multichromosomal genomes). The two methods allowing translocations require that the genomes H_1 and H_2 share the same set of chromosomal endpoints, but this can be taken care of by means of the addition of dummy endpoints, if necessary.

The algorithms all depend on a bicoloured graph $\mathcal{G}$ constructed from H_1 and H_2, for which the main ideas were introduced by Bafna and Pevzner (1996) and Kececioglu and Sankoff (1993). The details of this construction vary from model to model, due to the different ways chromosomal endpoints must be handled, but the general characteristics of the graph, as illustrated in figures 6.2 and 6.3, are the same and may be summarized as follows.

Graph $\mathcal{G}$: If gene x of H_1 has positive sign, replace it by the pair $x^t x^h$, and if it is negative, by $x^h x^t$. Then the vertices of $\mathcal{G}$ are just the x^t and the x^h for all genes x.

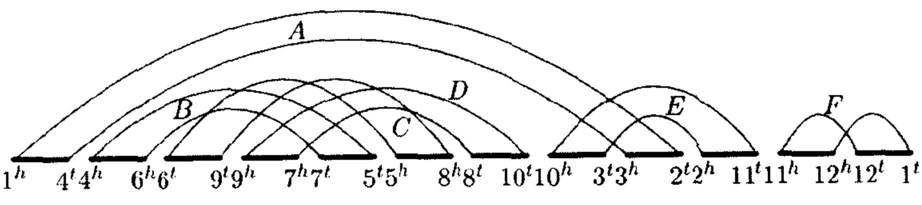


Figure 6.2
Graph $\mathcal{G}$ corresponding to circular genomes (i.e., first gene is adjacent to last gene) $H_1 = +1 + 4 - 6 + 9 - 7 + 5 - 8 + 10 + 3 + 2 + 11 - 12$ and $H_2 = +1 + 2 + 3 \cdots + 12$. A, B, C, D, E, and F are the 6 cycles of $\mathcal{G}$. $\{A, E\}$, $\{B, C, D\}$, and $\{F\}$ are the three components of $\mathcal{G}$.

Any two vertices that are adjacent in some chromosome in H_1, other than x^t and x^h from the same x, are connected by a black edge, and any two adjacent in H_2, by a gray edge. In the case of a single chromosome, the black edges may be displayed linearly according to the order of the genes in the chromosome. For a genome containing N chromosomes, N such linear orders are required; in the model allowing both reversals and translocations, however, the N orders are concatenated in each of the two genomes, so that we are again left with a single linear order.

Now, each vertex is incident to exactly one black and one gray edge, so that there is a unique decomposition of $\mathcal{G}$ into c disjoint cycles of alternating edge colours. By the *size of a cycle* we mean the number of black edges it contains. Note that c is maximized when $H_1 = H_2$, in which case each cycle has one black edge and one gray edge.

A rearrangement operation ρ, either a reversal or a translocation, is determined by the two black edges e and f where it "cuts" the current genome. Rearrangement operations may change the number of cycles, so that minimizing the number of operations can be seen in terms of increasing the number of cycles as fast as possible. Let $\mathcal{G}$ be a cycle graph, ρ a rearrangement operation, and $\Delta(c)$ the difference between the number of cycles before and after applying the operation ρ. Hannenhalli and Pevzner showed that $\Delta(c)$ may take on values 1, 0, or -1, in which cases they called ρ *proper*, *improper*, or *bad*, respectively. Roughly, an operation determined by two black edges in two different cycles will be bad, whereas one acting on two black edges within the same cycle may be proper or improper, depending on the type of cycle and the type of edges considered.

Key to the Hannenhalli-Pevzner approach are the graph components. Two cycles, say cycles 1 and 2, all of whose black edges are related by the same linear order (i.e.,

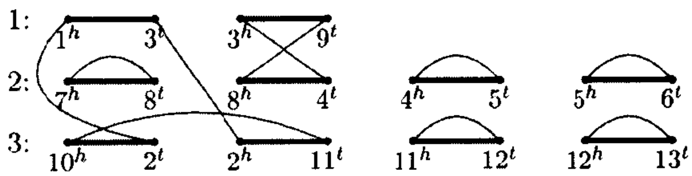


Figure 6.3
Graph $\mathcal{G}$ corresponding to genomes H_1, H_2, both with three chromosomes, where $H_1 = \{1\colon 1\ 3\ 9;\ 2\colon 7\ 8\ 4\ 5\ 6;\ 3\colon 10\ 2\ 11\ 12\ 13\}$ and $H_2 = \{1\colon 1\ 2\ 3\ 4\ 5\ 6;\ 2\colon 7\ 8\ 9;\ 3\colon 10\ 11\ 12\ 13\}$. All genes are signed "+." The edges that are on the same horizontal row of the graph correspond to a chromosome of H_1. Seven cycles are present. As no cycle of size > 1 is contained in one row, $\mathcal{G}$ does not contain any component. Both genomes have the same set of endpoints, so we can omit the first vertices (x^t for initial genes and x^h for terminal genes).

are on the same line), and containing gray edges that "cross," for example, gene i linked to gene j by a black edge (i.e., in H_1) in cycle 1, gene k linked to gene t by a black edge in cycle 2, but ordered i, k, j, t in H_2, are connected. For example, in figure 6.2, cycle A crosses cycle E by virtue of the four genes, 1, 4, 3 and 10. A component of $\mathcal{G}$ is a subset of the cycles (not consisting of a single cycle of size 1), built recursively from any of its cycles, at each step adding all the remaining cycles connected to any of those already in the construction. A component is termed *good* if it can be transformed to a set of cycles of size 1 by a series of proper operations, and *bad* otherwise. Bad components are called *subpermutations* in the translocations-only model, *hurdles* in the reversals-only model, and *knots* in the combined model. This property may be readily ascertained for each component by means of simple tests.

The Hannenhalli-Pevzner formulae for all three models may be summarized as follows:

$$d(H_1, H_2) = n(\mathcal{G}) - c(\mathcal{G}) + m(\mathcal{G}) + f(\mathcal{G})$$

where $d(H_1, H_2)$ is the minimum number of rearrangement operations (reversals and/or translocations), $n(\mathcal{G})$ is the number of black edges of $\mathcal{G}$, $c(\mathcal{G})$ is the number of cycles, $m(\mathcal{G})$ is the number of bad components, and $f(\mathcal{G})$ is a correction of size 0, 1, or 2 depending on the set of bad components.

6.5 Phylogenetic Analyses

Reconstruction of phylogeny may be approached through the application of generic methods (neighbor-joining, least-squares fitting, agglomerative clustering, etc.) to a distance matrix, independent of the nature of the data giving rise to the summary distances, or through ancestral inference methods (maximum likelihood, parsimony, etc.), where the tree shape is optimized simultaneously with the reconstruction of ancestral forms associated with nonterminal nodes, analogous to the input data associated with the terminal nodes. Distance matrices based on genomic distances have been used in traditional ways for phylogenetic reconstruction (Sankoff et al. 1992, 2000b), but approaches involving ancestral inference pose new analytical problems.

The problem of inferring ancestors may be decomposed into two aspects that must be solved simultaneously—finding the optimal shape, or topology, of the tree, and optimizing the ancestral reconstruction at each nonterminal node. Again, there are traditional search methods for optimal trees, but the reconstruction of ancestral genomes, given a fixed topology, is a new type of task, and it is on this question that we focus in this section.

6.5.1 The Median Problem

The solution of the *median problem* is of key importance in inferring the ancestral states in a phylogenetic tree. Given a distance d and three genomes A, B, and C, the median is a genome $M \in \mathcal{S}$, some set of eligible genomes, such that the sum $d(A, X) + d(B, X) - d(C, X)$ is minimal over $\mathcal{S}$ for $X = M$. Algorithms for finding the median can be used to reconstruct ancestors in a given phylogeny through the process of *steinerization*. Unfortunately, the median problem is NP-hard, even in the case of unique genes, for all known rearrangement distances d including signed inversion distance. Even heuristic approaches to this problem work well only for very small instances (cf. Hannenhalli et al. 1995; Sankoff et al. 1996; Caprara 2000).

Reversals Recall that reversal distance on signed genomes can be calculated in polynomial time; indeed, in only quadratic time. Can polynomial efficiency be extended to the median problem? The answer is no, as proved by Caprara (1999). Moreover, no heuristics for this problem have been shown to be reasonably effective for even moderate size instances.

Breakpoints For the breakpoint distance d, where $d(Y, Z)$ is the number of pairs of genes that are adjacent in genome Y but not in Z, the median problem is also NP-hard (Pe'er and Shamir 1998; Bryant 1998). Nevertheless, it can be solved in a relatively simple manner for three genomes A, B, and C, having the same gene content. Indeed, in this case, the problem can be reduced to the Traveling Salesman Problem (TSP) (Sankoff and Blanchette 1997).

For unsigned genomes, consider the complete graph Γ whose vertices are all the genes. For each edge gh, let $u(gh)$ be the number of times g and h are adjacent in the three genomes A, B, and C. Set $w(gh) = 3 - u(gh)$. Then the solution to TSP on (Γ, w) traces out an optimal genome M, because if g and h are adjacent in M, but not in A, for example, then they form a breakpoint in M.

For signed genomes, the reduction of the median problem to TSP must be somewhat different to take into account that we must specify whether the median genome contains $x^t x^h$ or $x^h x^t$, in the notation of section 6.4. Let Γ be a complete graph whose vertices include x^t and x^h for each gene x. For each pair of distinct genes x and y, let $u(xy)$ be the number of times x^h and y^t are adjacent in the genomes A, B, and C, and $w(xy) = 3 - u(xy)$. We also set $w(x^t x^h) = -Z$, where Z is large enough to assure that a minimum weight cycle must contain the edge $x^t x^h$.

Although the TSP is also NP-hard, the very rigid structure of the TSP graph Γ derived above may be exploited by developing specific heuristics (Sankoff and Blanchette 1997). There are a number of TSP algorithms and software packages applicable in particular contexts (Reinelt 1991). These allow us to find the median of

three genomes of size $n = 100$ in a matter of minutes (Sankoff and Blanchette 1998). Recently, we have developed a heuristic for this problem in the much more difficult case where the genomes do not have the same set of genes (Sankoff et al. 2000a, b).

Further work on these problems was done by Bryant (2000) and Pe'er and Shamir (2000).

6.5.2 Steinerization Algorithm

An optimal tree is one where the sum of the edge lengths is minimal, the length being defined as the number of breakpoints (or any other genomic distance) when the two genomes associated with the endpoints of the edge are compared. A binary unrooted tree may be decomposed into a number of overlapping median configurations. Each median consists of a nonterminal node together with its three adjacent nodes, terminal or nonterminal, and the three edges that join them. In an optimal tree, the genome reconstructed at each nonterminal node will be a solution to the median problem defined by its three neighbors. We can heuristically exploit this fact to reconstruct the ancestral genomes, starting with some reasonable initialization, and iterating the median algorithm on the list of nonterminal nodes until no improvement is found with any node. This may result in a local optimum, but sufficient repeated trials of the whole algorithm, with somewhat different initializations, should eventually indicate the best possible solution, or one very close to it. Blanchette et al. (1999) applied this method to animal mitochondrial genomes, and Cosner et al. (2000) to the chloroplast genomes of a family of flowering plants.

6.5.3 Probability-Based Models

The development of likelihood or other probability-based methods for phylogenetic inference from gene order data requires the prior probabilization of genome rearrangement models, which is much more difficult than modeling sequence divergence according to the Jukes-Cantor, Kimura, or the many other available parametrizations for nucleotide or amino acid residue substitutions, or even models allowing gaps. Sankoff and Blanchette (2000, 1999) gave a complete characterization of the evolution of gene adjacency probabilities for random reversals on unsigned circular genomes, as well as a recurrence in the case of reversals on signed genomes. Concepts from the theory of invariants developed for the phylogenetics of homologous gene sequences (Fu 1995) were used to derive a complete set of linear invariants for unsigned reversals, as well as for a mixed rearrangement model for signed genomes, though not for pure transposition or pure signed reversal models. The invariants are based on an extended Jukes-Cantor semigroup. The use of these invariants was illustrated by relating mitochondrial genomes from a number of invertebrate animals.

6.5.4 Reducing Gene Order Data to "Characters"

Gene adjacencies may be treated as characters in inferring a parsimony, maximum likelihood, or compatibility tree from gene order data (cf. Gallut et al. 2000; Cosner et al. 2000). The advantage of this is that it allows the use of existing phylogenetic software. The disadvantage is that the character sets it reconstructs at the ancestor nodes are generally incompatible with any gene order.

6.6 Gene Copies, Gene Families

There are a number of different ways in which duplicate genes can arise: tandem repeat through slippage during recombination, hybridization, polyploidization, duplication of all or part of a chromosome, gene conversion, and transposition of foreign genetic material, particularly horizontal (lateral) transfer from other genomes.

Analytical methods for genome rearrangement, predicated on the hypothesis that the gene order of two genomes are basically permutations of each other, eventually run into the problem of duplicate genes. It is no longer clear how to obtain the basic datum for rearrangement analysis: *caba* is not a permutation of *abc*. Complicating the situation further is the process of sequence divergence, whereby duplicate genes gradually become structurally and functionally differentiated; at some point they are no longer duplicates, but members of a gene family sharing some functional similarities as well as homology. Duplicate copies are also particularly prone to be lost, not so much through physical deletion but by becoming pseudogenes (nonfunctional ex-genes) through sequence divergence. This seems to happen much more rapidly in the case of individual gene duplication than in the context of whole or partial genome duplication (Nadeau and Sankoff 1997; Lynch and Conery 2000). It is in these contexts that the study of gene order is often forced to take account of the degree of similarity among different genes, and not to rely on a binary distinction between homologous and nonhomologous.

This section is structured according to the mechanism giving rise to duplicate genes. First, we discuss the doubling of the whole genome and the hybridization through fusion of two distinct genomes, then the processes of individual gene duplication, and finally horizontal transfer.

6.6.1 Genome Doubling

There is a difference between the duplication of single genes and processes that results in the doubling of large portions of a chromosome or even of the entire genome. In the latter case, not only is one copy of each gene free to evolve its own function (or to

lose its function, becoming a pseudogene and mutating randomly, eventually beyond recognition), but it can evolve in concert with any subset of the hundreds or thousands of other extra gene copies. Whole new physiological pathways may emerge, involving novel functions for many of these genes.

Evidence for the effects of genome duplication can be seen across the eukaryote spectrum, though it is always controversial (Ohno et al. 1968; Wolfe and Shields 1997; Postlethwait et al. 1998; Skrabanek and Wolfe 1998; Hughes 1999; Smith et al. 1999). Genome duplication and other mechanisms for combining two genomes (hybridization, allotetraploidization) are particularly prevalent in plants (Devos 2000; Parkin 2000; Paterson et al. 2000).

From the analytical point of view, partial or total genome duplication differs from mechanisms of duplication such as duplication-transposition, gene conversion, or horizontal transfer in that it conserves gene order within conserved segments, and this can facilitate the analysis of genomes descended from a duplicated genome.

A duplicated genome contains two identical copies of each chromosome, but through genome rearrangement parallel linkage patterns between the two copies are disrupted. Even after a considerable time, however, we can hope to detect a number of scattered chromosome segments, each of which has one apparent double, so that the two segments contain a certain number of paralogous genes in a parallel order. Similar patterns should be visible after hybridization through allotetraploidization (El-Mabrouk and Sankoff 1999a). The main methodological question addressed in this field is: How can we reconstruct some or most of the original gene order at the time of genome duplication or hybridization, based on traces conserved in the ordering of those duplicate genes still identifiable? Some of the contributions to this methodology include work by Skrabanek and Wolfe (1998); El-Mabrouk et al. (1998, 1999); El-Mabrouk and Sankoff (1999b), the latter applicable to single, circular chromosomal genomes, such as typical prokaryotes.

6.6.2 Multigene Families and Exemplar Distances

Implicit in definitions of rearrangement distances is that both genomes contain an identical set of genes and the one-to-one homologies (orthologies) between all pairs of corresponding genes in the two genomes have previously been established. As we have stressed, although this hypothesis of *unique genes* may be appropriate for some small genomes such as viruses and mitochondria, it is clearly unwarranted for divergent species where several copies of the same gene, or several homologous (paralogous) genes—a *multigene family*—may be scattered across a genome.

In a recent publication (Sankoff 1999), we formulated a generalized version of the genomic rearrangement problem, where each gene may be present in a number of

copies in the same genome. The central idea, based on a model of gene copy movement, is the deletion of all but one member of each gene family—its *exemplar*—in each of the two genomes being compared, so as to minimize some rearrangement distance d between the two reduced genomes thus derived. Therefore, the exemplar distance between two genomes X and Y is $e_d(X, Y) = \min d(X', Y')$, where the minimum is taken over all pairs of reduced genomes X' and Y' obtained by deleting all but one member of each gene family.

6.6.3 Duplication, Rearrangement, Reconciliation

The notion of exemplar distance takes on particular relevance in the phylogenetic context. Sankoff and El-Mabrouk (2000) investigated the problem of inferring ancestral genomes when the data genomes contain multigene families. We define a gene tree as a phylogenetic tree built from the sequences (according to some given method) of all copies of a gene g or all members of a gene family in all the genomes in the study. There are a number of techniques for inferring gene duplication events and gene loss events by projecting a gene tree T_g onto a "true" species tree T; this is known as *reconciliation* (e.g., Page and Cotton 2000).

We ask: Given

- a phylogenetic tree $\mathcal{T}$ on N species;
- their N genomes: strings of symbols belonging to an alphabet of size F;
- F gene trees, each T_g relating all occurrences of one symbol g in the N genomes;
- a distance d between two gene orders containing only unique genes,

the problem is to find, in each ancestral genome (internal node) of $\mathcal{T}$,

- its set of genes, as well as
- their relationships with respect to genes in the immediate ancestor,
- the order of these genes in the genome, and
- among each set of sibling genes (offspring of the same copy in the immediate ancestor), one gene, designated as the exemplar,

such that the sum of the branch lengths of the tree $\mathcal{T}$ is minimal. The length of the branch connecting a genome G to its immediate ancestor A is $e_d(G', A)$, where G' is the genome built from G by deleting all but the exemplar from each family.

6.6.4 Horizontal Transfer

Though molecular biologists have employed various filters, such as testing for aberrant codon usage, to detect horizontally transferred genes within genomes, formal

methods for this phenomenon are just beginning to be developed. Hallett and Lagergren (2000) have investigated a problem where a number of (conflicting) gene trees are to be mapped to a species tree in such a way as to minimize the number of transfer events implied.

6.7 Future Directions

Analyzing the complexity of simple measures of genomic distance, such as those involving transpositions, and devising exact and heuristic algorithms for them remains a rich source of research problems for theoretical computer scientists. But perhaps more important is the development of parametrized models of gene order divergence and the inferential apparatus, both combinatorial and statistical, necessary to apply these in a meaningful way to prokaryotic and eukaryotic genomes. Some of the parameters are those pertaining to type of rearrangement (e.g., inversion, transposition, translocation), the size of the chromosomal segment(s) involved, and the initial and final chromosomal positions of the segment (e.g., centromeric, telomeric).

Interesting problems in combinatorial probability await researchers modeling gene order evolution, and this may turn out to be of particular importance for prokaryotic genomes (Sankoff 2000), where conserved gene "clusters" may play a role analogous to that of conserved segments in higher eukaryotic genomes.

Phylogenetic analysis based on gene order is a difficult field, but one that is increasingly important. Progress on the median problem or other approaches to recovering ancestral gene order is crucial.

The integration of gene duplication and gene family studies with genome rearrangement theory is a new but potentially powerful way of resolving ambiguity, nonuniqueness, and other questions of interpretation within each of these fields separately. Practical algorithms in this area would seem to depend on progress in phylogenetic analysis, but there is much theoretical work to be done as well.

In contrast to many areas of computational biology, genome rearrangement theory is driven less by the immediate preoccupations of molecular biologists and geneticists, and even less by commercial applicability, though there has been considerable investment by researchers in domesticated plants and animals (cf. Devos 2000; Womack 2000). Thus the emergence of new problems depends on the intrinsic interest provoked by comparative genomic data now being produced, and by the analytical approaches favoured by those that study them.

One area of great importance that has not been analyzed from the perspectives of the work reviewed here is that of chromosomal rearrangements in oncology. There are rich data on the extensive and diverse karyotypes, and the rearrangement events

underlying them, in a range of cancers (Mitelman et al. 1997). These neoplastic patterns, arising *somatically*, have not been examined from the modeling and algorithmic points of view discussed in this chapter. The analysis of tumor genome rearrangements, in comparison to the *genetic* rearrangements we have been reviewing, including pathological, otherwise deleterious and relatively neutral ones with evolutionary consequence, could well contribute to our understanding of cancer cell biology.

Acknowledgments

Research has been supported in part by grants from the Natural Sciences and Engineering Research Council of Canada. D.S. is a fellow, and NE-M a scholar, of the Program in Evolutionary Biology of the Canadian Institute for Advanced Research.

References

Andersson, S. G. E., and Eriksson, K. (2000). Dynamics of gene order structures and genomic architectures. In *Comparative Genomics*, Sankoff, D. and Nadeau, J. H., eds., 267–280. Dordrecht, NL: Kluwer Academic Press.

Bafna, V., Beaver, D., Fürer, M., and Pevzner, P. A. (2000). Circular permutations and genome shuffling. In *Comparative Genomics*, Sankoff, D. and Nadeau, J. H., eds., 199–206. Dordrecht, NL: Kluwer Academic Press.

Bafna, V., and Pevzner, P. A. (1996). Genome rearrangements and sorting by reversals. *SIAM J. Comput.* 25(2): 272–289.

Bafna, V., and Pevzner, P. A. (1998). Sorting by transposition. *SIAM J. Discrete Math.* 11(2): 224–240.

Berman, P., and Hannenhalli, S. (1996). Fast sorting by reversals. In *Proceedings of the Seventh Annual Symposium on Combinatorial Pattern Matching (CPM '96)*, Hirschberg, D. and Myers, G., eds., vol. 1075 of *Lecture Notes in Computer Science*, 168–175. Berlin: Springer.

Blanchette, M., Kunisawa, T., and Sankoff, D. (1996). Parametric genome rearrangement. *Gene* 172: GC11–GC17.

Blanchette, M., Kunisawa, T., and Sankoff, D. (1999). Gene order breakpoint evidence in animal mitochondrial phylogeny. *J. Mol. Evol.* 49: 193–203.

Bryant, D. (1998). The complexity of the breakpoint median problem. Technical Report CRM-2579, Centre de recherches mathématiques, Université de Montréal.

Bryant, D. (2000). A lower bound for the breakpoint phylogeny problem. In *Proceedings of the Eleventh Annual Symposium on Combinatorial Pattern Matching (CPM 2000)*, Giancarlo, R. and Sankoff, D., eds., vol. 1848 of *Lecture Notes in Computer Science*, 235–247. Berlin: Springer.

Caprara, A. (1997). Sorting by reversals is difficult. In *Proceedings of the First Annual International Conference on Computational Molecular Biology (RECOMB 97)*, 75–83. New York: ACM Press.

Caprara, A. (1999). Formulations and hardness of multiple sorting by reversals. In *Proceedings of the Third Annual International Conference on Computational Molecular Biology (RECOMB 99)*, Istrail, S., Pevzner, P. A., and Waterman, M. S., eds., 84–93. New York: ACM.

Caprara, A. (2000). Practical solution for the reversal median problem. Manuscript.

Caprara, A., Lancia, G., and Ng, S. K. (2000). Fast practical solution of sorting by reversal. In *Proceedings of the 11th ACM-SIAM Symposium on Discrete Algorithms*, 12–21. New York: ACM.

Christie, D. A. (1999). *Genome Rearrangement Problems.* Ph.D. dissertation, Univeristy of Glasgow.

Cosner, M. E., Jansen, R. K., Moret, B. M. E., Raubeson, L. A., Wang, L.-S., Warnow, T., and Wyman, S. (2000). An empirical comparison of phylogenetic methods on chloroplast gene order data in Campanulaceae. In *Comparative Genomics*, Sankoff, D. and Nadeau, J. H., eds., 99–121. Dordrecht, NL: Kluwer Academic Press.

Dalevi, D., Eriksen, N., Eriksson, K., and Andersson, S. (2000). Genome comparison: The number of evolutionary events separating *C. pneumoniae* and *C. trachomatis*. Technical report, University of Uppsala.

DasGupta, B., Jiang, T., Kannan, S., Li, M., and Sweedyk, E. (1998). On the complexity and approximation of syntenic distance. *Discrete Appl. Math.* 88(1–3): 59–82.

Devos, K. M. (2000). Comparative genetics: From hexaploid wheat to arabidopsis. In *Comparative Genomics*, Sankoff, D. and Nadeau, J. H., eds., 411–423, Dordrecht, NL: Kluwer Academic Press.

El-Mabrouk, N. (2000). Genome rearrangement by reversals and insertions/deletions of contiguous segments. In *Proceedings of the Eleventh Annual Symposium on Combinatorial Pattern Matching (CPM 2000)*, Giancarlo, R. and Sankoff, D., eds., vol. 1848 of *Lecture Notes in Computer Science*, 222–234. Berlin: Springer.

El-Mabrouk, N., Bryant, B., and Sankoff, D. (1999). Reconstructing the predoubling genome. In *Proceedings of the Third Annual International Conference on Computational Molecular Biology (RECOMB '99)*, Istrail, S., Pevzner, P. A., and Waterman, M. S., eds., 154–163. New York: ACM Press.

El-Mabrouk, N., Nadeau, J. H., and Sankoff, D. (1998). Genome halving. In *Proceedings of the Ninth Annual Symposium on Combinatorial Pattern Matching (CPM '98)*, Farach-Colton, M., ed., vol. 1448 of *Lecture Notes in Computer Science*, 235–250. Heidelberg: Springer Verlag.

El-Mabrouk, N., and Sankoff, D. (1999a). Hybridization and genome rearrangement. In *Proceedings of the Tenth Annual Symposium on Combinatorial Pattern Matching (CPM 1999)*, Crochemore, M. and Paterson, M., eds., vol. 1645 of *Lecture Notes in Computer Science*, 78–87. Berlin: Springer Verlag.

El-Mabrouk, N., and Sankoff, D. (1999b). On the reconstruction of ancient doubled circular genomes using minimum reversals. In *Genome Informatics 1999*, Asai, K., Miyano, S., and Takagi, T., eds., 83–93. Tokyo: Universal Academy Press.

Ferretti, V., Nadeau, J. H., and Sankoff, D. (1996). Original synteny. In *Proceedings of the Seventh Annual Symposium on Combinatorial Pattern Matching (CPM '96)*, Hirschberg, D. and Myers, G., eds., vol. 1075 of *Lecture Notes in Computer Science*, 159–167. Berlin: Springer.

Fu, Y. X. (1995). Linear invariants under Jukes' and Cantor's one-parameter model. *J. Theoret. Biol.* 173: 339–352.

Gallut, C., Barriel, V., and Vignes, R. (2000). Gene order and phylogenetic information. In *Comparative Genomics*, Sankoff, D. and Nadeau, J. H., eds., 123–132. Dordrecht, NL: Kluwer Academic Press.

Gu, Q.-P., Iwata, K., Peng, S., and Chen, Q.-M. (1997). A heuristic algorithm for genome rearrangements. In *Proceedings of the Eighth Workshop on Genome Informatics 1997*, Miyano, S. and Takagi, T., eds., 268–269. Tokyo: Universal Academy Press.

Hallett, M. T., and Lagergren, J. (2000). Efficient algorithms for horizontal gene transfer problems. Manuscript.

Hannenhalli, S. (1996). Polynomial-time algorithm for computing translocation distance between genomes. *Discrete Appl. Math.* 71: 137–151.

Hannenhalli, S., Chappey, C., Koonin, E. V., and Pevzner, P. A. (1995). Genome sequence comparison and scenarios for gene rearrangements: A test case. *Genomics* 30: 299–311.

Hannenhalli, S., and Pevzner, P. (1995). Transforming men into mice (polynomial algorithm for genomic distance problem). In *Proceedings of the IEEE 36th Annual Symposium on Foundations of Computer Science*, 581–592. Los Alamitos, CA: IEEE Computer Society.

Hannenhalli, S., and Pevzner, P. A. (1999). Transforming cabbage into turnip (polynomial algorithm for sorting signed permutations by reversals). *J. ACM* 48: 1–27.

Hughes, A. L. (1999). *Adaptive Evolution of Genes and Genomes*. New York: Oxford University Press.

Jackson, R. (1957). New low chromosome number for plants. *Science* 126: 1115–1116.

Kaplan, H., Shamir, R., and Tarjan, R. E. (2000). A faster and simpler algorithm for sorting signed permutations by reversals. *SIAM J. Comput.* 29: 880–892.

Kececioglu, J., and Sankoff, D. (1993). Exact and approximation algorithms for the inversion distance between two permutations. In *Proceedings of the Fourth Annual Symposium on Combinatorial Pattern Matching (CPM '93)*, Apostolico, A., Crochemore, M., Galil, Z., and Manber, U., eds., vol. 684 of *Lecture Notes in Computer Science*, 87–105. Berlin: Springer.

Kececioglu, J., and Sankoff, D. (1994). Efficient bounds for oriented chromosome inversion distance. In *Proceedings of the Fifth Annual Symposium on Combinatorial Pattern Matching (CPM '94)*, Crochemore, M. and Gusfield, D., eds., vol. 807 of *Lecture Notes in Computer Science*, 162–176. Berlin: Springer.

Kececioglu, J., and Sankoff, D. (1995). Exact and approximation algorithms for sorting by reversals, with application to genome rearrangement. *Algorithmica* 13: 180–210.

Kececioglu, J. D., and Ravi, R. (1995). Of mice and men: Algorithms for evolutionary distance between genomes with translocations. In *Proceedings of the Sixth ACM-SIAM Symposium on Discrete Algorithms*, 604–613. New York: ACM Press.

Kleinberg, J., and Liben-Nowell, D. (2000). The syntenic diameter of the space of N-chromosome genomes. In *Comparative Genomics*, Sankoff, D. and Nadeau, J. H., eds., 185–197. Dordrecht, NL: Kluwer Academic Press.

Liben-Nowell, D. (1999). On the structure of syntenic distance. In *Proceedings of the Tenth Annual Symposium on Combinatorial Pattern Matching (CPM '99)*, Crochemore, M. and Paterson, M., eds., vol. 1645 of *Lecture Notes in Computer Science*, 43–56. Berlin: Springer.

Lima-de Faria, A. (1980). How to produce a human with 3 chromosomes and 1000 primary genes. *Hereditas* 93: 47–73.

Lynch, M., and Conery, J. S. (2000). The evolutionary fate and consequences of duplicated genes. *Science* 1151–1155.

McAllister, B. F. (2000). Fixation of chromosomal rearrangements. In *Comparative Genomics*, Sankoff, D. and Nadeau, J. H., eds., 19–27. Dordrecht, NL: Kluwer Academic Press.

McLysaght, A., Seoighe, C., and Wolfe, K. H. (2000). High frequency of inversions during eukaryote gene order evolution. In *Comparative Genomics*, Sankoff, D. and Nadeau, J. H., eds., 47–58, Dordrecht, NL: Kluwer Academic Press.

Meidanis, J., and Dias, Z. (2000). An alternative algebraic formalism for genome rearrangements. In *Comparative Genomics*, Sankoff, D. and Nadeau, J. H., eds., 213–233. Dordrecht, NL: Kluwer Academic Press.

Mitelman, F., Mertens, F., and Johansson, B. (1997). A breakpoint map of recurrent chromosomal rearrangements in human neoplasia. *Nature Genet.* 15: 417–474.

Nadeau, J. H., and Sankoff, D. (1997). Comparable rates of gene loss and functional divergence after genome duplications early in vertebrate evolution. *Genetics* 147: 1259–1266.

Nadeau, J. H., and Sankoff, D. (1998). Counting on comparative maps. *Trends Genet.* 14: 495–501.

Ohno, S., Wolf, U., and Atkin, N. B. (1968). Evolution from fish to mammals by gene duplication. *Hereditas* 59: 169–187.

Page, R. D. M., and Cotton, J. A. (2000). GENETREE: A tool for exploring gene family evolution. In *Comparative Genomics*, Sankoff, D. and Nadeau, J. H., eds., 525–536. Dordrecht, NL: Kluwer Academic Press.

Parkin, I. (2000). Unraveling crucifer genomes through comparative mapping. In *Comparative Genomics*, Sankoff, D. and Nadeau, J. H., eds., 425–537, Dordrecht, NL: Kluwer Academic Press.

Paterson, A. H., Bowers, J. E., Burow, M. D., Draye, X., Elsik, C. G., Jiang, C.-X., Katsar, C. S., Lan, T.-H., Lin, Y.-R., Ming, R., and Wright, R. J. (2000). Comparative genomics of plant chromosomes. In *Comparative Genomics*, Sankoff, D. and Nadeau, J. H., eds., 439–457, Dordrecht, NL: Kluwer Academic Press.

Pe'er, I., and Shamir, R. (1998). The median problems for breakpoints are NP-complete. Electronic Colloquium on Computational Complexity Technical Report 98-071. http://www.eccc.uni-trier.de/eccc.

Pe'er, I., and Shamir, R. (2000). Approximation algorithms for the median problem in the breakpoint model. In *Comparative Genomics*, Sankoff, D. and Nadeau, J. H., eds., 225–241. Dordrecht, NL: Kluwer Academic Press.

Postlethwait, J. H., Yan, Y.-L., Gates, M. A., Horne, S., Amores, A., Brownlie, A., Donovan, A., Egan, E. S., Force, A., Gong, Z., Goutel, C., Fritz, A., Kelsh, R., Knapik, E., Liao, E., Paw, B., Ransom, D., Singer, A., Thomson, T., Abduljabbar, T. S., Yelick, P., Beier, D., Joly, J.-S., Larhammar, D., Rosa, F., Westerfield, M., Zon, L. I., and Talbot, W. S. (1998). Vertebrate genome evolution and the zebrafish gene map. *Nature Genet.* 18: 345–349.

Reinelt, G. (1991). *The Traveling Salesman—Computational Solutions for TSP Applications.* Berlin: Springer Verlag.

Sankoff, D. (1992). Edit distance for genome comparison based on nonlocal operations. In *Proceedings of the Third Annual Symposium on Combinatorial Pattern Matching (CPM '92)*, Apostolico, A., Crochemore, M., Galil, Z., and Manber, U., eds., vol. 644 of *Lecture Notes in Computer Science*, 121–135. Berlin: Springer.

Sankoff, D. (1999). Genome rearrangements with gene families. *Bioinformatics* 15: 909–917.

Sankoff, D. (2000). Short inversions and conserved gene clusters. Manuscript.

Sankoff, D., and Blanchette, M. (1997). The median problem for breakpoints in comparative genomics. In *Computing and Combinatorics, Proceeedings of COCOON '97*, Jiang, T. and Lee, D. T., eds., vol. 1276 of *Lecture Notes in Computer Science*, 251–263. Berlin: Springer.

Sankoff, D., and Blanchette, M. (1998). Multiple genome rearrangement and breakpoint phylogeny. *J. Comput. Biol.* 5: 555–570.

Sankoff, D., and Blanchette, M. (1999). Phylogenetic invariants for genome rearrangements. *J. Comput. Biol.* 6: 431–445.

Sankoff, D., and Blanchette, M. (2000). Comparative genomics via phylogenetic invariants for Jukes-Cantor semigroups. In *Stochastic Models: A Conference in Honour of Professor Donald A. Dawson*, Gorostiza, L. and Ivanoff, B., eds., vol. 26 of *Canadian Mathematical Society Conference Proceedings Series*, 399–418. Providence, RI: American Mathematical Society.

Sankoff, D., Bryant, D., Deneault, M., Lang, B. F., and Burger, G. (2000a). Early eukaryote evolution based on mitochondrial gene order breakpoints. *J. Comput. Biol.* 7: 521–535.

Sankoff, D., Cedergren, R., and Abel, Y. (1990). Genomic divergence through gene rearrangement. In *Molecular Evolution: Computer Analysis of Protein and Nucleic Acid Sequences*, Doolittle, R. F., ed., vol. 183 of *Methods in Enzymology*, 428–438. Academic Press.

Sankoff, D., Deneault, M., Bryant, D., Lemieux, C., and Turmel, M. (2000b). Chloroplast gene order and the divergence of plants and algae, from the normalized number of induced breakpoints. In *Comparative Genomics*, Sankoff, D. and Nadeau, J. H., eds., 89–98. Dordrecht, NL: Kluwer Academic Press.

Sankoff, D., and El-Mabrouk, N. (2000). Duplication, rearrangement and reconciliation. In *Comparative Genomics*, Sankoff, D. and Nadeau, J. H., eds., 537–550. Dordrecht, NL: Kluwer Academic Press.

Sankoff, D., and Goldstein, M. (1989). Probabilistic models of genome shuffing. *Bull. Math. Biol.* 51: 117–124.

Sankoff, D., Leduc, G., Antoine, N., Paquin, B., Lang, B. F., and Cedergren, R. J. (1992). Gene order comparisons for phylogenetic inference: Evolution of the mitochondrial genome. *Proc. Natl. Acad. Sci. USA* 89(14): 6575–6579.

Sankoff, D., Sundaram, G., and Kececioglu, J. (1996). Steiner points in the space of genome rearrangements. *Int. J. Found. Comput. Sci.* 7: 1–9.

Skrabanek, L., and Wolfe, K. H. (1998). Eukaryote genome duplication—where's the evidence? *Curr. Opinion Genet. Devel.* 8: 694–700.

Smith, N. G. C., Knight, R., and Hurst, L. D. (1999). Vertebrate genome evolution: A slow shuffle or a big bang? *BioEssays* 21: 697–703.

Sturtevant, A. H., and Novitski, E. (1941). The homologies of chromosome elements in the genus Drosophila. *Genetics* 26: 517–541.

Walter, M. E., Dias, Z., and Meidanis, J. (1998). Reversal and transposition distance of linear chromosomes. In *String Processing and Information Retrieval: A South American Symposium (SPIRE '98)*, 96–102. Los Alamitos, CA: IEEE Computer Society.

Watterson, G., Ewens, W., Hall, T., and Morgan, A. (1982). The chromosome inversion problem. *J. Theoret. Biol.* 99: 1–7.

Wolfe, K. H., and Shields, D. C. (1997). Molecular evidence for an ancient duplication of the entire yeast genome. *Nature* 387: 708–713.

Womack, J. E. (2000). The essential role of comparative maps in livestock genomics. In *Comparative Genomics*, Sankoff, D. and Nadeau, J. H., eds., 401–409, Dordrecht, NL: Kluwer Academic Press.

7 Compressing DNA Sequences

Ming Li

7.1 Overview

With the imminent completion of the Human Genome project and the fast increase of many complete genomes of prokaryotes and eukaryotes, fundamental questions regarding the characteristics of these sequences arise (Koonin 1999; Wooley 1999), the first of which is how to compare genomes. We introduce an effective general tool for such questions: compression programs for DNA sequences. We will first review some compression algorithms for biological sequences, and then switch our attention to the question of how to use such compression programs to compare genomes.

Why are we interested in compressing DNA sequences? From a strictly mathematical point of view, compression implies understanding and comprehension (Li and Vitanyi 1997); from a more utilitarian point of view, as we will demonstrate soon, compression is a great tool for genome comparison and for studying various properties of genomes.

Life represents order. It is neither chaotic nor random (Li and Vitanyi 1997). In other words, DNA sequences, which encode life, should be compressible. There is also strong biological evidence that supports this claim: it is well known that DNA sequences, especially in higher eukaryotes, contain many (approximate) tandem repeats; it is also well known that many essential genes (like rRNAs) have many copies; it is believed that there are only about a thousand basic protein folding patterns; it also has been conjectured that genes duplicate themselves sometimes for evolutionary or simply for "selfish" purposes. All these reasons give more concrete support that DNA sequences should be reasonably compressible. However, such regularities are often blurred by random mutation, translocation, cross-over, and reversal events, as well as sequencing errors.

The compression of DNA sequences is a very difficult task (Curnow and Kirwood 1989; Grumback and Tahi 1994; Rivals et al. 1995; Lanctot et al. 2000; Chen et al. 2000). DNA sequences consist of only four nucleotide bases $\{a, c, g, t\}$; two bits are enough to store each base. However, if one applies standard compression software such as the Unix "compress," "bzip2," and "gzip," or the MS-DOS archive programs "winzip" and "arj," they all *expand* the file with more than two bits per base, as shown in table 7.1, although all this compression software uses universal compression algorithms. These software tools are designed for English text compression (Bell et al. 1990), but the regularities in DNA sequences are subtler.

One may treat compressibility study as the ultimate generalization of the simpler (and fruitful) biological studies such as G-C contents of various species. More sophis-

Table 7.1
Compression measured in bits per base

Sequence	Size (bases)	compress	arith-2	Biocom-press-2	GenCom-press-1	GenCom-press-2
MTPACGA	100314	2.116	1.873	1.875	1.861	1.861
MPOMTCG	186608	2.202	1.966	1.938	1.898	1.898
CHNTXX	155844	2.187	1.934	1.617	1.614	1.614
CHMPXX	121024	2.075	1.837	1.685	1.669	1.670
HUMGHCSA	66495	2.194	1.938	1.307	1.092	1.097
HUMHBB	73323	2.195	1.918	1.877	1.813	1.814
HUMHDABCD	58864	2.230	1.943	1.877	1.800	1.809
HUMDYSTROP	38770	2.233	1.924	1.926	1.924	1.923
HUMHPRTB	56737	2.202	1.929	1.907	1.826	1.830
VACCG	191737	2.167	1.898	1.761	1.761	1.761
HEHCMVCG	229354	2.213	1.965	1.848	1.847	1.847

ticated studies on DNA sequences will give us a deeper understanding about the nature of these sequences. Different regions on a genome, different genes, different species may have different compression ratios. Such differences may imply, for example, different mutation rates in different genes (Lanctot et al. 2000).

We will also discuss conditional compressibility where one compresses one sequence given another sequence as free information. Intuitively, conditional compressibility implies some relatedness between two sequences. However, directly using it results in an incorrect measure, which is not symmetric. We define a proper distance to measure how much information two DNA sequences or two genomes share. As an example, we will show how our compression programs can be used to construct whole genome trees.

Unless otherwise mentioned, we will use the lower case letters u, v, w, x, y to denote finite strings over the alphabet $\{a, c, g, t\}$. $|u|$ denotes the length (i.e., number of characters) of u. u_i is the i-th character of u. $u_{i:j}$ is the substring of u from position i to position j. The first character of u is u_0. Thus $u = u_{0:|u|-1}$. We use ϵ to denote empty string and $|\epsilon| = 0$.

7.2 GenCompress: A DNA Sequence Compression Program

Grumbach and Tahi (1993, 1994) proposed two lossless compression algorithms for DNA sequences, namely *Biocompress* and *Biocompress-2*, in the spirit of the Ziv and Lempel (1977) data compression method. *Biocompress-2* detects exact repeats and complementary palindromes located earlier in the target sequence, and then encodes them by repeat length and the position of a previous repeat occurrence. In addition,

Biocompress-2 also uses arithmetic coding of order 2 if no significant repetition is found. In fact, the difference between *Biocompress* and *Biocompress-2* is the addition of order-2 arithmetic coding.

Rivals et al. (1995) give another compression algorithm, *Cfact*, which searches the longest exact matching repeat in an entire sequence using a suffix tree data structure. The idea of *Cfact* is basically the same as *Biocompress-2* except that *Cfact* is a two-pass algorithm. It builds the suffix tree in the first pass. In the encoding phase, the repetitions are coded with guaranteed gain; otherwise, two-bit per base encoding is used. This is similar to the codeword encoding condition in *Biocompress-2* except that the order-2 arithmetic coding is not used in *Cfact*. É. Rivals et al. (1997) also designed a compression algorithm as a tool to detect the approximate tandem repeats in DNA sequences.

Lempel and Ziv proposed two algorithms (Ziv and Lempel 1977; Lempel and Ziv 1978) to compress universal data sequences. These are dictionary based compression algorithms that rely on exact repeats. The Lempel-Ziv algorithms can be viewed as having two components: the first component is to parse the input data sequence into variable-length strings based on the history of the dictionary. The second component is to replace the variable-length prefix by a proper binary codeword—concatenation of these codewords yields the encoder's output sequence in response to the input data sequence. We follow the same framework and generalize it to approximate matching for DNA sequences.

GenCompress (Chen et al. 2000) achieves significantly higher compression ratios than either *Biocompress-2* or *Cfact*. Such improvement is key to its application in genome comparison. *GenCompress* is a one-pass algorithm. It proceeds as follows: For input w, assume that a part of it, say v, has already been compressed, and the remaining part is u, that is, $w = vu$. *GenCompress* finds an "optimal prefix" of u such that it approximately matches some substring in v so that this prefix of u can be encoded economically. After outputting the code of this prefix, remove the prefix from u, and append it to the suffix of v. Continue the process till $u = \epsilon$.

We adopt the following constraint in *GenCompress* to limit the search. If the number of edit operations (insert, delete, replace) located in any substring of length k in the prefix s of u for an edit operation sequence $\lambda(s,t)$ is not larger than a threshold value b, we say that $\lambda(s,t)$ satisfies the *condition* $C = (k, b)$ for compression. In *GenCompress*, we only search for approximate matches that satisfy condition C. This way we limit our search space. Experiments show that setting C to $(k, b) = (12, 3)$ gives good results.

We defined a *compression gain function* G in order to evaluate if a particular approximate repeat provides profit in the encoding.

Table 7.2
Compression measured bits per base

Sequence	Size (bases)	LZW 15	arith-2	Cfact	GenCompress-1	GenCompress-2
atatsgs	9647	2.237	1.951	1.785	1.664	1.673
atef1a23	6022	2.297	1.994	1.585	1.541	1.540
atrdnaf	10014	2.300	2.009	1.814	1.789	1.786
atrdnai	5287	2.239	1.994	1.468	1.419	1.410
hsg6pdgen	52173	2.168	1.937	1.928	1.785	1.800
xlxfg512	19338	2.084	1.923	1.490	1.376	1.385
mmzp3g	10833	2.244	1.953	1.911	1.854	1.857
celk07e12	58949	2.108	1.912	1.713	1.597	1.605

Let input $w = vu$, where v has already been processed. Given $G(s, t, \lambda)$ and C, the *optimal prefix* is a prefix s of u such that $G(s, t, \lambda)$ is maximized over all λ and t such that t is a substring of v and λ is an edit transcription from t to s satisfying condition C.

The algorithm carefully finds the optimal prefix, and uses order-2 Arithmetic encoding (Nelson 1991; Bell et al. 1990) whenever needed. *GenCompress* also detects the approximate complemented palindrome in DNA sequences.

Some standard benchmark data has been used (Grumback and Tahi 1994). These standard sequences (available at GeneBank 1999) come from a variety of sources and include the complete genomes of two mitochondria, MPOMTCG, PANMTPACGA (also called MIPACGA); two chloroplasts, CHNTXX and CHMPXX (also called MPOCPCG); five sequences from humans, HUMGHCSA, HUMHBB, HUMHDABCD, HUMDYSTROP, HUMHPRTB; and finally the complete genome from the two viruses, VACCG and HEHCMVCG (also called HS5HCMVCG).

The compression ratios of *GenCompress*, as well as those of *Biocompress-2* and some other compression algorithms, are presented in table 7.1. The comparison of *GenCompress* with *Cfact* is presented in table 7.2, using the data from Rivals et al. (1995). Note that although *Cfact* looks for the best matches globally, whereas our *GenCompress* only searches for the best approximate match from the current prefix to the part of the text seen so far. *GenCompress* has a much better compression ratio than *Cfact*. From these experiments, it is clear that approximate matching plays a key role in finding similarities or regularities in DNA sequences.

In conclusion, the compression results of *GenCompress* for DNA sequences indicate that our method based on approximate matching is more effective than others. *GenCompress* is able to detect more regularities and achieve better compression results.

7.3 GTAC: A DNA Sequences Entropy Estimator

In many applications, we are more interested in estimating DNA sequence entropy than really achieving the final compression. The final compression size in bits usually overestimates the true entropy.

There have been several attempts to characterize the entropy of DNA. One of the most common approaches is to estimate the probability of n-tuples for large n, and use this value to compute the block entropy (entropy of n-tuples). One problem with this approach is that it converges too slowly, requiring an exponentially large dataset. Even though genome databases are large and growing larger, the values that are obtained systematically overestimate the entropy due to the finite sample effect, and must be corrected. Several researchers address this problem and have developed methods to correct it, such as Liò et al. (1996), and Schmitt and Herzel (1997).

Farach et al. (1994) developed a novel algorithm to estimate the entropy of DNA sequences called a match length entropy estimator. This algorithm was used to test the differences between the entropy of introns and exons. Farach et al. also proved that their algorithm was universal, that is, that the entropy estimate will approach the true entropy as the size of the sequence increases, but only under the assumption that the sequence is generated by a Markov process.

Loewenstern and Yianilos (1999) developed CDNA, a program that estimates the entropy of DNA sequences. The motivation for CDNA comes from the observation that naturally occurring DNA sequences contain many more near repeats than

Table 7.3
Comparison of entropy values in bits per symbol

Sequence name	Sequence length	UNIX compress	Biocompress-2	CDNA compress	GTAC
PANMTPACGA	100314	2.12	1.88	1.85	1.74
MPOMTCG	186609	2.20	1.94	1.87	1.78
CHNTXX	155844	2.19	1.62	1.65	1.53
CHMPXX	121124	2.09	1.68		1.58
SCCHRIII	315339	2.18	1.92	1.94	1.82
HUMGHCSA	66495	2.19	1.31	0.95	1.10
HUMHBB	73308	2.20	1.88	1.77	1.73
HUMHDABCD	58864	2.21	1.88	1.67	1.70
HUMDYSTROP	38770	2.23	1.93	1.93	1.81
HUMHPRTB	56737	2.20	1.91	1.72	1.72
VACCG	191737	2.14	1.76	1.81	1.67
HEHCMVCG	229354	2.20	1.85		1.74

Note that UNIX-compress and Biocompress-2 are lossless compression algorithms. CDNA and GTAC are entropy estimators.

Table 7.4
Features of various entropy estimators

Algorithm	Universal	Linear run time	Entropy estimate
UNIX compress	yes	yes	worst
Match length	limited	yes	
Biocompress-2	yes	yes	3rd best
CDNA	no	no	2nd best
GTAC	yes	yes	best

would be expected by chance. Two parameters that CDNA uses to capture the inexact matches are w, which represents the substring size, and h, which represents the Hamming distance. These parameters are used to create a panel of predictive experts $p_{w,h}$, each with differing values of w and h. CDNA then learns the weightings of these various experts, using Expectation Maximization, so that their predictive ability is maximized when combined into a single prediction. CDNA uses everything to the left of a nucleotide as the learning set to predict its value. The average over-all position is calculated and is the value recorded in table 7.3.

We have developed a new entropy estimator of DNA sequences, GTAC (Lanctot et al. 2000), which is based on the idea of Kieffer and Yang (1999) regarding the design and analysis of grammar based codes, and which recognizes the repeats and reverse complement property of DNA sequences. This entropy estimator is universal as it does not assume any source model and works for any individual sequence. This entropy estimator is proved to be universal. Table 7.4 compares GTAC with several entropy estimators.

Kieffer and Yang (1999) recently put forth a new type of lossless source code called a grammar based code, and developed a new universal lossless source coding theory. In this theory, a grammar based code has the structure shown in figure 7.1.

The idea is as follows. Given a sequence x, we generate a particular context-free grammar G_x such that one can recover x from G_x. For example, here is G_x for $x = aataaatgcaatatatatgc$.

$$S \rightarrow BADBCCD$$

$$A \rightarrow aa$$

$$B \rightarrow At$$

$$C \rightarrow at$$

$$D \rightarrow Cgc$$

Then we use an arithmetic coder to encode G_x.

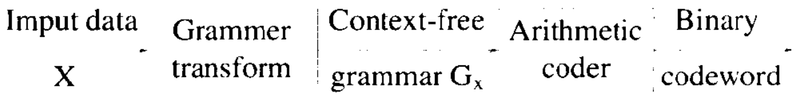


Figure 7.1
Structure of a grammar-based code.

To characterize the resulting compression rate, let $\omega(G_x)$ be the sequence obtained by concatenating the righthand side of all production rules of G_x in some order and then deleting the first appearance of each variable (because they can be uniformly replaced by one special symbol and, during decoding, can be recovered by their order of apparences). Let

$$H(G_x) = \sum_s n(s) \log \frac{|\omega(G_x)|}{n(s)} \tag{7.1}$$

where the summation is taken over all variables and terminal symbols in G_x, $n(s)$ denotes the number of times the variable (or terminal symbol) s appears in $\omega(G_x)$, and $|\omega(G_x)|$ denotes the length of the sequence $\omega(G_x)$. Also, in formula (7.1), the logarithm is relative to base 2, and the convention $0 \log \infty = 0$ is adopted. In terms of the terminology in Kieffer and Yang (1999), the quantity $H(G_x)$ is called the unnormalized entropy of the grammar G_x. For the CFG G_x shown in example 1, $\omega(G_x) = BCDaaAtatCgc$ and $H(G_x) = 34.26$. The following theorem, proved by Kieffer and Yang (1999), characterizes the resulting compression rate.

THEOREM 7.1. According to arithmetic coding or enumerative coding, one can assign a uniquely decodable binary codeword $B(G_x)$ to each admissible CFG G_x (or its equivalent form) such that

$$|B(G_x)| = f(G_x) + H(G_x) \tag{7.2}$$

where $|B(G_x)|$ denotes the length of the binary codeword $B(G_x)$, and $f(G_x)$ represents the overhead paid to the universality of grammar based codes. In (7.2), $f(G_x)$ is negligible compared to $H(G_x)$ and is upper bounded, by

$$f(G_x) \leq 5|G_x| + 4$$

where $|G_x|$ denotes the total length of the righthand sides of all production rules of G_x.

Because it can be shown that $f(G_x) \leq 5\, G_x| \leq O(|x|/\log|x|)$, $H(G_x)/|x|$ goes to the actual per-bit-entropy of x (Kieffer and Yang 1999). This justifies the use $H(G_x)$ as the entropy of x.

In order to estimate $H(G_x)$, we need to first construct G_x. Our task is to repeatedly find a longest match, and then replace it by a nonterminal and create a new rule in the grammar G_x. Such a task, done trivially, easily costs $\Omega(n^3)$. Using a suffix tree data structure (Gusfield 1997), and by carefully maintaining changes on the suffix tree, a linear time algorithm is possible for computing G_x (Lanctot et al. 2000). Note that the linear time algorithm is important here because the sequences we are compressing are often of many megabases. Table 7.3 compares the entropy estimates on the standard benchmark data using various programs. GTAC gives the lowest entropy estimates in most cases.

Together with Jonathan Badger, we have performed some preliminary experiments with interesting consequences. One such experiment is concerned with coding and noncoding regions in *E. coli*. Around 90 percent of the genome of higher eukaryotes is noncoding, whereas about 15 percent of the genome of *E. coli* is noncoding. If noncoding regions have a definite role, they may be more regular than coding regions, which would support the conjecture that noncoding regions in prokaryotes are not junk. Our results confirmed this hypothesis. When comparing coding and noncoding regions of *E. coli*, we found the following entropy values:

- 1.85 bits/symbol for coding regions (4,090,525 bases)
- 1.80 bits/symbol for noncoding regions (640,039 bases).

7.4 A New Distance Measure and Whole Genome Comparison

As we accumulate an enormous amount of nucleotide data (doubling every 18 months), one exciting challenge facing bioinformatics researchers is to provide tools to analyze such data (Koonin 1999; Wooley 1999). Entropy estimators provide a convenient tool for some aspects of such a task.

Given two sequences x and y, DNA or otherwise, we (Chen et al. 2000; Li et al. 2000) have defined a distance measure between x and y as:

$$d(x, y) = 1 - \frac{K(x) - K(x \mid y)}{K(xy)}$$

where $K(x)$ is the Kolmogorov complexity of the string x, that is, the the length in bits of the shortest program causing a standard universal computer to compute x as its unique output. $K(x \mid y)$ is the conditional Kolmogorov complexity, assuming the above program has y as extra free information. See Li and Vitanyi (1997) for a comprehensive introduction to the subject and its applications. The numerator of the

fraction is the amount of information y knows about x, which is equal to the amount of information x knows about y, by a theorem in Kolmogorov complexity (Li and Vitanyi 1997), to within a logarithmic error. This is called mutual algorithmic information, which is not itself a distance and does not satisfy the triangle inequality; however, $d(x, y)$ as defined above does (Li et al. 2001), ranging from a minimum of 0 when $x = y$, to a maximum of 1 when x and y are independent strings of equal or differing length. Distance function $d(x, y)$ is not only well-defined, it is also *universal* in the sense that if any distance measure uncovers some similarity between two sequences, so will d. Of course, it is well known that K is an uncomputable function (Li and Vitanyi 1997); accordingly, we used our program *GenCompress*, discussed in section 7.2, to heuristically approximate it. Notice that *GTAC* is an unconditional entropy estimator, and at this point it cannot be used to do conditional entropy estimation. The corresponding conditional entropy theory and program is under development.

Different approaches for comparing genomes or general sequences have been proposed (Grumback and Tahi 1994; Varre et al. 1998; Snel et al. 1999; Boore and Brown 1998; Fitz-Gibbon and House 1999; Bennett et al. 1998). Grumback and Tahi (1994) proposed to use conditional compression. Using ideas of Kolmogorov complexity, Varre, Delahaye, and Rivals (Varre et al. 1998) defined "transformation distance." Essentially, this can be regarded as conditional compression using biologically related operations. Although very attractive, both of these measures are not symmetric, and hence cannot be used as distance in general. This situation can in fact be remedied by using Information Distance as defined by Bennett et al. (1998) (see theorem 8.3.1 in Li and Vitanyi 1997). However, information distance (Bennett et al. 1998; Li and Vitanyi 1997), although symmetric, is also not a right measure in this case. The Information Distance is not suitable in genome comparison because it would overpunish long deletions.

Biologists (Snel et al. 1999; Boore and Brown 1998; Fitz-Gibbon and House 1999) proposed to use more involved and laborious methods like counting the number of shared genes in two genomes or comparing the ordering of the genes. These distances theoretically may be regarded as special cases of our $d(x, y)$, because each of these proposed measures may be regarded simply as one way of compression.

These distances, together with G+C content, edit distance, and reversal and rearrangement distances (Kececioglu and Sankoff 1995; Hannenhalli and Pevzner 1995; Nadeau and Sankoff 1998) compare genomes using only partial genome information, and with a pre-assumed model of similarity, whereas our new distance uses all genome information and makes no assumption of evolutionary model, at least in theory.

To demonstrate that this theory is applicable, we have performed many experiments on genomes. One is described here. It has been debated which two of the three main groups of placental mammals are more closely related: primates, ferungulates, and rodents. This is because by the maximum likelihood method, some proteins support the (ferungulates, (primates, rodents)) grouping, whereas other proteins support the (rodents, (ferungulates, primates)) grouping (Cao et al. 1998). Cao et al. (1998) aligned 12 concatenated mitochondrial proteins from the following species: rat (*Rattus norvegicus*), house mouse (*Mus musculus*), grey seal (*Halichoerus grypus*), harbor seal (*Phoca vitulina*), cat (*Felis catus*), white rhino (*Ceratotherium simum*), horse (*Equus caballus*), finback whale (*Balaenoptera physalus*), blue whale (*Balaenoptera musculus*), cow (*Bos taurus*), gibbon (*Hylobates lar*), gorilla (*Gorilla gorilla*), human (*Homo sapiens*), chimpanzee (*Pan troglodytes*), pygmy chimpanzee (*Pan paniscus*), orangutan (*Pongo pygmaeus*), and Sumatran orangutan (*Pongo pygmaeus abelii*), using opossum (*Didelphis virginiana*), wallaroo (*Macropus robustus*), and platypus (*Ornithorhynchus anatinus*) as the outgroup, and built the maximum likelihood tree to confirm the grouping (rodents, (primates, ferungulates)). Using the complete mitochondrial genomes of these species, we approximated our new distance $d(x, y)$ between each pair of species x and y. We then constructed a tree (figure 7.2) using the neighbor joining (Saitou and Nei 1987) program in Adachi and Hasegawa's MOLPHY package (Adachi et al. 1996). The tree is identical to the maximum likelihood tree of Cao et al. (1998). Because neighbor-joining is sometimes distrusted (Hillis et al. 1994; Kuhner and Felsenstein 1994), to further corroborate this grouping we applied our own hypercleaning program (Berry et al. 2000) to the same distance matrix and obtained the same tree. The hypercleaning program constructs an evolutionary tree using the edges best supported by all possible four taxa subtrees (commonly called "quartets"). Thus using the new information-theoretic distances derived from the *complete* mtDNA genomes we have re-confirmed the hypothesis of (rodents, (primates, ferungulates)). The distance matrix can be found at the author's homepage: http://www.math.uwaterloo.ca/~mli/distance.html.

The simple asymmetric measure $K(x \mid y)$ leads to a wrong tree using the same data and programs, as expected (data not shown). The gene order (Boore and Brown 1998) and gene content (Snel et al. 1999; Fitz-Gibbon and House 1999) approaches, although yielding symmetric distances, have the disadvantage of requiring laborious human analysis of the sequences and also are unlikely to provide enough information to distinguish closely related species such as the above data set. Note that this complete process is fully automatic and utilizes the information contained in noncoding regions in addition to the information contained in the genes.

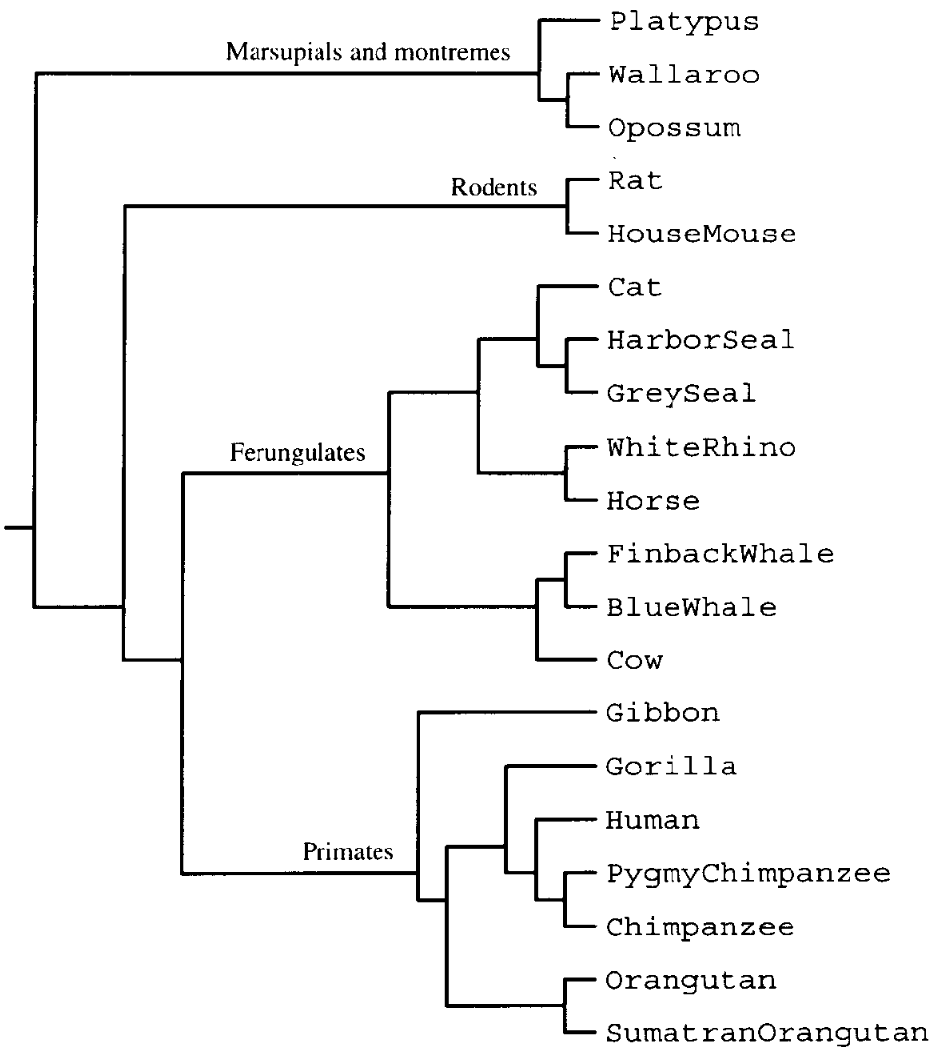


Figure 7.2
The evolutionary tree built from complete mammalian mtDNA sequences.

Of course, the question is whether this method will in fact work on complete genomes. We have performed a small-scale experiment with the following seven complete genomes (GeneBank 1999):

- Archaea Bacteria: *Archaeoglobus fulgidus* (u1), *Pyrococcus abyssi* (u2), *Pyrococcus horikoshii* OT3 (u3)
- Bacteria: *Escherichia coli* K-12 MG1655 (u4), *Haemophilus influenzae Rd* (u5), *Helicobacter pylori* 26695 (u6); *Helicobacter pylori*, strain J99 (u7).

The resulting distance matrix, using $100(1 - d(x, y))$, is given in table 7.5 and the corresponding evolutionary tree is in figure 7.3.

Table 7.5
Distance $d(u, v)$ between all pairs u, v

Sequence	*u1*	*u2*	*u3*	*u4*	*u5*	*u6*	*u7*
u1		4226821[b]	4226743	4228299	4228411	4228356	4228392
2178400[a]		0.018326[c]	0.019550	−0.000548	−0.002399	−0.001765	−0.002259
u2	3443540		3391432	3445299	3445242	3445257	3445264
1765118	0.023072		0.797546	0.000089	0.000988	0.000812	0.000705
u3	3362288	3310372		3364086	3363996	3364031	3364045
1738505	0.023055	0.794383		−0.000391	0.000617	0.000109	−0.000109
u4	8920179	8920362	8920294		8914204	8919249	8919224
4639221	0.000373	−0.001084	−0.000537		0.048760	0.008160	0.008371
u5	3440205	3440216	3440229	3434165		3439033	3439068
1830138	0.000274	0.000145	−0.000044	0.049059		0.018303	0.017776
u6	3079174	3078992	3079021	3077924	3077935		1226333
1667867	−0.002217	0.000307	−0.000140	0.009068	0.016523		43.069863
u7	3075330	3075285	3075238	3074059	3073952	1219515	
1643831	−0.001314	−0.000782	−0.000062	0.009796	0.019680	43.171044	

[a] Number of bases in the input sequence.
[b] Number of bits of conditionally compressed file between u_i and u_j.
[c] $100\dfrac{K(v) - K(v|u)}{K(uv)}$.

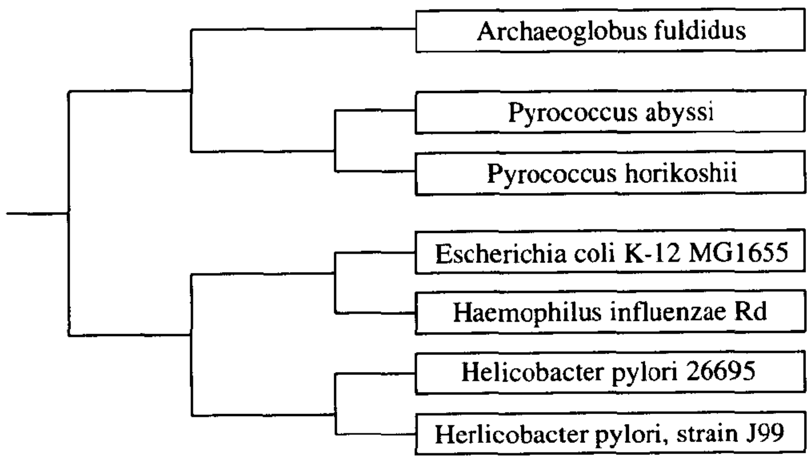


Figure 7.3
The phylogeny for seven genomes derived from table 7.5.

7.5 Discussion

We hope this new methodology and the automatic tool we have developed could serve as an alternative approach to comparing genomes and constructing whole genome trees. It perhaps would serve as a quick and dirty initial way to compare genomes. This new method for whole genome comparison and phylogeny requires neither gene identification nor any human intervention; in fact, it is totally automatic, and mathematically well-founded. It works when there are no agreed upon evolutionary models, as further demonstrated by the successful construction of a chain letter phylogeny (Bennett et al. 2000) and when individual gene trees do not agree (Cao et al. 1998; Doolittle 1999; Lawrence and Ochman 1998), as is the case for genomes.

However, there are many questions that remain to be answered and experiments to be performed. It is important to design a good conditional entropy estimator for DNA sequences. Only with a better and fast conditional entropy estimator can this method become effective. It is important to perform more experiments with this method to identify the range of data such a method works for. One potential objection to our method is that if noncoding regions are usually junk, then are we measuring junk? For prokaryotes, at least, this is not the case, as our experiments (performed by J. H. Badger, using *GTAC*) show that coding regions for *E. coli* have higher entropy than noncoding regions of *E. coli*. It is also interesting to extend our study here to other domains, such as program comparison. Currently we are working on this project.

Acknowledgments

This work was supported in part by City University of Hong Kong Grant No. 7000875, NSERC Research Grant OGP0046506, CITO, a CGAT grant, the Steacie Fellowship, and an NSF ITR grant. I am grateful to X. Chen and S. Kwong for implementing *GenCompress*, and K. Lanctot and E. H. Yang for their work on *GTAC*. I would like to thank my other coauthors for their work on several projects discussed in this paper: J. Badger, C. H. Bennett, P. Kearney, T. Jiang, B. Ma, J. Tsang, H. C. Wang, H. Y. Zhang. I would also like to thank both referees—one of them especially has painstakingly helped me to correct many errors.

References

Adachi, J., and Hasegawa, M. (1996). MOLPHY version 2.3: Programs for molecular phylogenetics based on maximum likelihood. *Comput. Sci. Monogr. Inst. Stat. Math* 28: 1–150.

Apostolico, A., and Fraenkel, A. S. (1987). Robust transmission of unbounded strings using Fibonacci representations. *IEEE Trans. Inform. Theory* IT-33(2): 238–245.

Bell, T. C., Cleary, J. G., and Witten, I. H. (1990). *Text Compression*. New York: Prentice Hall.

Bennett, C. H., Gács, P., Li, M., Vitányi, P., and Zurek, W. (1998). Information distance. *IEEE Trans. Inform. Theory* 44:4 (July): 1407 1423. (Also in *STOC'93*.)

Bennett, C. H., Li, M., Ma, B. Linking chain letters. To appear in *Scientific American*.

Berry, V., Bryant, D., Jiang, T., Kearney, P., Li, M., Wareham, T., and Zhang, H. (2000). A practical algorithm for recovering the best supported edges in an evolutionary tree. *Proc. 11th ACM-SIAM Sypm. Disc. Alg.*, ACM Press, New York, NY.

Boore, J. L., and Brown, W. M. (1998). Big trees from little genomes: Mitochondrial gene order as a phylogenetic tool. *Curr. Opin. Genet. Dev.* 8(6): 668–674.

Cao, Y., Janke, A., Waddell, P. J., Westerman, M., Takenaka, O., Murata, S., Okada, N., Pääbo, S., and Hasegawa, M. (1998). Conflict among individual mitochondrial proteins in resolving the phylogeny of Eutherian orders. *J. Mol. Evol.* 47: 307–322.

Chen, X., Kwong, S., and Li, M. (2000). A compression algorithm for DNA sequences and its application in genome comparison. In Proc. Genome Informatics Workshop (GIW '99), Tokyo, Japan, Dec. 1999.

Curnow, R., and Kirkwood, T. (1989). Statistical analysis of deoxyribonucleic acid sequence data—a review. *J. Royal Statist. Soc.* 152: 199–220.

Doolittle, W. F. (1999). Phylogenetic classification and the universal tree. *Science* 284: 2124–2129.

Farach, M., Noordewier, M., Savari, S., Shepp, L., Wyner, A., and Ziv, A. (1994). On the entropy of DNA: Algorithms and measurements based on memory and rapid convergence. *Proceedings of the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, 48–57. ACM, New York, NY.

Fitz-Gibbon, S. T., and House, C. H. (1999). Whole genome-based phylogenetic analysis of free-living microorganisms. *Nucl. Acids Res.* 27: 4218–4222.

Gardner, E. J., Sinnoms, M. J., and Snustad, D. P. (1991). *Principles of Genetics*, 8th ed. Wiley, New York, NY.

Grumbach, S., and Tahi, F. (1994). A new challenge for compression algorithms: Genetic sequences. *Journal of Information Processing and Management* 30(6): 875–866.

Grumbach, S., and Tahi, F. (1993). Compression of DNA sequences. In *Proc. IEEE Symp. on Data Compression* 340–350. Los Alamitos, CA: IEEE.

Gusfield, D. (1997). *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*. New York: Cambridge University Press.

Hannenhalli, S., and Pevzner, P. (1995). Transforming cabbage into turnip. *Proc. 27th ACM Symp. Theory of Computing*, pp. 178–189, New York: ACM Press.

Hillis, D. M., Huelsenbeck, J. P., and Swofford, D. L. (1994). Hobgoblin of phylogenetics? *Nature* 369: 363–364.

Kececioglu, J., and Sankoff, D. (1995). Exact and approximation algorithms for the inversion distance. *Algorithmica* 13: 180–210.

Kieffer, J., and Yang, E. Grammar based codes: A new class of universal lossless source codes. Submitted for journal publication.

Koonin, E. V. (1999). The emerging paradigm and open problems in comparative genomics. *Bioinformatics* 15: 265–266.

Kuhner, M. K., and Felsenstein, J. (1994). A simulation comparison of phylogeny algorithms under equal and unequal evolutionary rates. *Mol. Biol. Evol.* 11: 459–468.

Lanctot, K., Li, M., and Yang, E. H. (2000). Estimating DNA sequence entropy. *Proc. 11th ACM-SIAM Sypm. Disc. Alg.*, 409–418. New York: ACM.

Lawrence, L. G., and Ochman, H. (1998). Molecular archaeology of the *Escherichia coli* genome. *Proc. Natl. Acad. Sci. USA* 95: 9413–9417.

Liò, P., Politi, A., Buiatti, M., and Ruffo, S. (1996). High statistics block entropy measures of DNA sequences. *Journal of Theoretical Biology* 180: 151–160.

Lempel, A., and Ziv, J. (1978). Compression of individual sequences via variable-rate coding. *IEEE Trans. Inform. Theory* IT-24: 530–536.

Li, M., Badger, J. H., Chen, X., Kwong, S., Kearney, P., and Zhang, H. (2001). An information based distance and its application to whole mitochondrial genome phylogeny. To appear in *Bioinformatics* 17: 149–154.

Li, M., and Vitányi, P. (1997). *An Introduction to Kolmogorov Complexity and its Applications*. 2nd ed. Berlin: Springer.

Loewenstern, D., and Yianilos, P. (1999). Significantly lower entropy estimates for natural DNA sequences. *Journal of Computational Biology* 6: 125–142.

Milosavljevic, A., and Jurka, J. (1993). Discovery by minimal length encoding: A case study in molecular evolution. *Machine Learning* 12: 69–87.

National Center for Biotechnology Information, Entrez Nucleotide Query, http://www.ncbi.nlm.nih.gov/htbin-post/Entrez/query?db=n_s.

Nelson, M. (1991). *The Data Compression Book*. New York: M&T Publishing Inc.

Nadeau, J. H., and Sankoff, D. (1998). Counting on comparative maps. *Trends Genet.* 14: 495–501.

Rivals, É., Delahaye, J-P., Dauchet, M., and Delgrange, O. (1995). A guaranteed compression scheme for repetitive DNA sequences. LIFL Lille I University, technical report IT-285.

Rivals, É., Delgrange, O., Delahaye, J-P., Dauchet, M., Delorme, M-O., Hénaut, A., and Ollivier, E. (1997). Detection of significant patterns by compression algorithms: The case of approximate tandem repeats in DNA sequences. *CABIOS* 13(2): 131–136.

Saitou, N., and Nei, M. (1987). The neighbor-joining method: A new method for reconstructing phylogenetic trees. *Mol. Biol. Evol.* 4: 406–425.

Schmitt, A., and Herzel, H. (1997). Estimating the entropy of DNA sequences. *Journal of Theoretical Biology* 188: 369–377.

Snel, B., Bork, P., and Huynen, M. A. (1999). Genome phylogeny based on gene content. *Nat. Genet.* 21(1): 108–110.

Varre, J-S., Delahaye, J-P., and Rivals, E. (1998). The transformation distance: A dissimilarity measure based on movements of segments. German conference on bioinformatics, Koel, Germany.

Wooley, J. C. (1999). Trends in computational biology: A summary based on a RECOMB plenary lecture, 1999. *J. Comput. Biol.* 6(3/4): 459–474.

Ziv, J., and Lempel, A. (1977). A universal algorithm for sequential data compression. *IEEE Trans. Inform. Theory* 23(3): 337–343.

III DATA MINING AND PATTERN DISCOVERY

8 Linkage Analysis of Quantitative Traits

Shizhong Xu

8.1 Introduction

Quantitative traits are usually defined as traits that have a continuous phenotypic distribution (Falconer and Mackey 1996; Lynch and Walsh 1998), such as the growth rate of plants. Variances of these traits are often controlled by the segregation of many loci; therefore, quantitative traits are also called polygenic traits. Another characteristic of quantitative traits is that environmental effects can play a large role in the variation of phenotypic distribution. The polygenic nature and the ability of being modified by the environment make the study of the genetic basis for quantitative traits more difficult than that for monogenic traits. There is another class of traits that appear to be qualitative phenotypically but have a polygenic genetic background. These traits are called threshold traits, such as disease susceptibility. Threshold characters are usually studied using statistical methods similar to those used in the study of quantitative traits. Therefore, the study of the genetic basis for threshold traits is also covered by quantitative genetics.

Because of the polygenic nature, traditional methods of quantitative genetics that use only the phenotypic and pedigree information cannot separate the effects of individual loci but study the collective effect of all. With the rapid development of molecular technology, a large number of highly polymorphic molecular markers (DNA variants) can be generated with ease. Most molecular markers are functionally neutral, but they normally obey the laws of Mendelian inheritance. Therefore, the relative relationship of the markers in the genome (called the marker map) can be reconstructed using observed recombinant events. The joint segregating patterns of markers under a given marker map, in conjunction with phenotypic and pedigree information, provides additional information about the genetic basis of quantitative traits, including the number of quantitative trait loci (QTL), the mode of gene action and the effects of each QTL, and the locations of these QTL along the chromosomes. A complete description of the properties of these individual loci is called the genetic architecture of quantitative traits. Study of the genetic architecture of quantitative traits using molecular markers is called linkage analysis or QTL mapping.

QTL mapping could lead to several useful applications. First, it could improve the efficiency of selective breeding. Second, transgenic technology might be applied to quantitative traits. Third, the identification of alleles causing predisposition to common multifactorial diseases could lead to improved methods of prevention. Fourth, quantitative genetics theory will be made more realistic when the number and prop-

erties of the genes are known, and more realistic theories will improve our understanding of evolution (Falconer and Mackey 1996).

8.2 Overview of QTL Mapping Statistics

Linkage disequilibrium is the foundation for QTL mapping, as it creates marker-trait associations, with different marker genotypes having different expected values for characters influenced by QTL linked to these markers (Weir 1996). Therefore, the first step of QTL mapping is to create a population with linkage disequilibrium. The simplest method for creating such a population is to make a cross between two inbred lines. For mapping purposes, crosses between inbred lines have the fewest complications. The progeny from such crosses display maximum disequilibrium. Using F_1 parents, a variety of populations, such as backcross and F_2, can be generated for mapping. However, many organisms in nature are outbred. Creating inbred lines may not be easy for some species due to temporal, economical, or biological limitations. Mapping QTL in such outbred populations must take advantage of existing data. Such data, however, do not usually show strong linkage disequilibrium. Therefore, we have to take a family based approach, where each family is considered as a small mapping population. If multiple families are used, results are usually combined and mapping is actually conducted within families. The theoretical basis of within family mapping is that linkage disequilibrium is always expected within a family, even if equilibrium is expected across families in the population.

Analyses of data obtained from different populations often require different statistical methods. The statistical methods reviewed in this section are only applicable to populations derived from the crosses of inbred lines. Developing methods for mapping outbred populations with pedigree data is the main scheme of this study and will be described in section 8.4.

There are many different designs of line crossing experiments. However, I will use a backcross (BC), the simplest design of line cross, as an example to discuss the methods. This design starts with the cross of two inbred lines. The hybrid is called the F_1, which is then crossed back to one of the inbred parents to generate a collection of BC individuals. QTL mapping is performed in this BC family.

8.2.1 Least Squares

The least squares (LS) method is derived by treating each marker as a candidate QTL. Define the genotypes of the two inbred parents at the locus of interest by A_1A_1 and A_2A_2, respectively. The F_1 hybrid will have a genotype of A_1A_2. Assume that the

F_1 is crossed back to the A_1A_1 parent. The two possible genotypes in the BC family are A_1A_1 and A_1A_2, and assuming additivity, with corresponding genotypic values of a and 0, respectively. Note that the genotypic value of A_1A_2 has been arbitrarily assigned zero and thus a becomes the substitution effect of allele A_1 relative to allele A_2. Define $z_j = 1$ if individual j has inherited the A_1 allele from the F_1 parent, that is, j is of genotype A_1A_1, and $z_j = 0$ if j inherited the A_2 allele from the F_1 parent. The phenotypic value of individual j can be described by the following linear model,

$$y_j = \mu + z_j a + \varepsilon_j \tag{8.1}$$

where μ is the mean and ε_j is the residual error with an assumed $N(0, \sigma_\varepsilon^2)$ distribution. This is a typical linear model with z_j as the regressor and a as the regression coefficient. The estimate of a is obtained through the ordinary least squares analysis. A simple t or F test can be applied to test the significance of the estimated a.

In reality, the genotype of the QTL is not observable, and thus z_j is missing. What we can observe is the segregation of a marker locus linked to the QTL. Assume that the recombination fraction between the marker and the QTL is c. Define the two possible genotypes of the marker in the BC family as M_1M_1 and M_1M_2, respectively. Similar to z_j, let us further define the indicator variable for the marker genotype by m_j, where $m_j = 1$ for M_1M_1 and $m_j = 0$ for M_1M_2. Substituting z_j by its expectation conditional on m_j, we get

$$y_j = \mu + E(z_j \mid m_j)a + e_j \tag{8.2}$$

where $E(z_j \mid m_j) = b_0 + b_1 m_j = c + (1 - 2c)m_j$. Therefore,

$$y_j = \mu^* + m_j a^* + e_j \tag{8.3}$$

where $\mu^* = \mu + b_0 a = \mu + ca$ and $a^* = b_1 a = (1 - 2c)a$. When a marker genotype is used in place of the QTL genotype, we actually estimate and test a^* instead of a. Note that a^* is a confounded effect of the size of the QTL and the recombination fraction. Either $1 - 2c = 0$ or $a = 0$ will cause $a^* = 0$. The first term, $1 - 2c$, is the correlation coefficient between z and m, which ranges from 0 as $c = 1/2$ (no linkage) to 1 as $c = 0$ (complete linkage).

Lander and Botstein (1989) proposed to use two markers simultaneously, one on each side of the QTL, to infer the distribution of z_j. Because the two flanking markers define an interval covering a range of possible locations of the putative QTL, the method is referred to as interval mapping. Lander and Botstein (1989) used a maximum likelihood (ML) method to estimate and test a, which will be discussed in the next section. Only the LS method for interval mapping will be

described in this section. The LS method was independently developed by Haley and Knott (1992) and Martinez and Curnow (1992). It is a simple extension of the single marker analysis with $E(z_j \mid m_j)$ substituted by $E(z_j \mid m_{1j}, m_{2j})$, where m_{1j} and m_{2j} are indicator variables for the left and right flanking markers, respectively. Because z_j is a Bernoulli variable, $E(z_j \mid m_{1j}, m_{2j}) = \Pr(z_j = 1 \mid m_{1j}, m_{2j})$. Let c_i (for $i = 1, 2$) be the recombination fraction between marker i and the QTL, and c_{12} be the recombination fraction between the two markers. The conditional probabilities are $\Pr(z_j = 1 \mid 1, 1) = (1 - c_1)(1 - c_2)/(1 - c_{12})$, $\Pr(z_j = 1 \mid 1, 0) = (1 - c_1)c_2/c_{12}$, $\Pr(z_j = 1 \mid 0, 1) = c_1(1 - c_2)/c_{12}$ and $\Pr(z_j = 1 \mid 0, 0) = c_1c_2/(1 - c_{12})$. Letting $p_j = \Pr(z_j = 1 \mid m_{1j}, m_{2j})$ and substituting z_j by p_j, we have a re-expressed linear model of

$$y_j = \mu + p_j a + e_j \tag{8.4}$$

The usual least squares method and the t or F test are then applied to estimate and test a.

The recombination fraction between the two markers, c_{12}, is assumed to be known (estimated prior to QTL mapping). Given c_{12}, c_2 is a simple function of c_1, as implied in $c_{12} = c_1(1 - c_2) + (1 - c_1)c_2$. This relationship provides a simple scheme of chromosome scanning. We start from the marker in one end of the chromosome and test each putative position until we reach the marker in the other end of the chromosome. For every putative position, only the flanking markers are used to infer the conditional expectation of z_j. While walking along the chromosome, we plot the test statistic value against the position and form a test statistic profile. The regions where the peaks of the profile occur are candidate locations for QTL along the chromosome.

8.2.2 Maximum Likelihood

When p_j is used in place of z_j, the residual error in the original model (ε_j) is replaced by e_j accordingly. In fact, e_j has a mixture of two normal distributions. This is due to the fact that an individual cannot take a genotypic value of $p_j a$; rather, the genotypic value will be either a or 0, with a probability of p_j or $1 - p_j$, respectively. If the value is 0, the distribution of e_j will be $N(0, \sigma_\varepsilon^2)$; otherwise, the it will be $N(a, \sigma_\varepsilon^2)$. So the distribution is actually $p_j N(a, \sigma_\varepsilon^2) + (1 - p_j)N(0, \sigma_\varepsilon^2)$. The variance of e_j has been inflated as a result of neglecting the mixed nature of the distribution. In addition, the substitution of z_j by p_j has violated the assumption of homogenous residual variance in ordinary least squares. Xu (1995) showed that

$$Var(e_j) = p_j(1 - p_j)a^2 + \sigma_\varepsilon^2 \tag{8.5}$$

The amount of inflation is determined jointly by the uncertainty of z_j, that is, $Var(z_j) = p_j(1 - p_j)$, and the size of the QTL, a. Therefore, the simple least squares

method is only approximately valid when the markers are closely linked to the QTL or the QTL has a small effect. The maximum likelihood method of interval mapping (Lander and Botstein 1989) actually takes into account the mixture distribution and has been proven to be optimal.

Let $\phi(y_j \mid z_j)$ be a normal density with mean $\mu + z_j a$ and variance σ_ε^2, the exact form of the likelihood function,

$$L(\mu, a, \sigma_\varepsilon^2) = \prod_{j=1}^{N} [\phi(y_j \mid 1) p_j + \phi(y_j \mid 0)(1 - p_j)] \tag{8.6}$$

where N is the number of individuals in the mapping population. The optimal property of ML is achieved at the expense of losing the explicit solution for the estimated QTL effect. Fortunately, with a simple modification of the least squares program, the Expectation Maximization (EM) iteration algorithm (Dempster et al. 1977) can be easily implemented.

The test statistic for $a = 0$ under the ML framework is the likelihood ratio statistic,

$$\xi = -2[\ln(L_0) - \ln(L_1)] \tag{8.7}$$

where $L_1 = L(\hat{\mu}, \hat{a}, \hat{\sigma}_\varepsilon^2)$ is the likelihood value under the full model and $L_0 = L(\hat{\mu}, 0, \hat{\sigma}_\varepsilon^2)$ is the likelihood value under the restricted model, that is, under $a = 0$.

8.2.3 Weighted Least Squares

The simple LS method fails to take into account two facts in the QTL model: (1) the mixture distribution of the residual error, and (2) the heterogeneous residual variances across individuals. Xu (1998) proposed a weighted least squares method to eliminate the second problem. The weight given to the jth individual is

$$w_j = \left[\frac{a^2}{\sigma_\varepsilon^2} p_j(1 - p_j) + 1\right]^{-1} \tag{8.8}$$

Because the weight is a function of the parameters, iterations are required. The iteration starts with $w_j = 1$ for all j's. The solutions at the first iteration are actually the ordinary least squares estimates. The weight is then updated using the estimated a and σ_ε^2 in the previous iteration. We then use the updated weight to reestimate the parameters. The iteration process continues until a given convergence criterion has been reached. Because recalculations of the weight are required, the method is called iteratively reweighted least squares (IRWLS). The result of this method is almost identical to that of ML. Unlike the EM iterations in ML, however, IRWLS normally takes only a few iterations to converge.

8.2.4 Bayes Method

Although LS and ML are still the main statistical methods used in QTL mapping, considerable attention has been paid to the study of Bayesian mapping. For simple genetic models, such as with no more than one QTL on each chromosome, the LS and ML methods are adequate, but they are not optimal for handling multiple QTL models. In particular, they do not allow the inference of the number of QTL, one of the important parameters in QTL mapping. The composite interval mapping approach (Jansen 1993; Zeng 1994) was developed for multiple QTL. However, it is still a one-dimension algorithm and thus provides a partial solution for the multiple QTL problem. Recently, a stepwise regression approach has been suggested to search for the optimal number of QTL (Kao et al. 1999). Unfortunately, the method has numerous unsolved problems. The Bayesian method provides a possible optimal solution for this problem (Satagopan et al. 1996; Hoeschele et al. 1997; Sillanpää and Arjas 1998). The Bayesian method implemented via the Markov chain Monte Carlo (MCMC) technique can solve relatively more complicated models. The cost of the MCMC, however, is high because of the intensive computation required in the sampling process. This had previously prohibited the use of the Bayesian method. The barrier has now disappeared due to the ever growing power of computers.

Bayesian mapping allows the use of prior knowledge of QTL parameters. In the situation where no prior information is available, one can choose a flat (uninformative) prior. In most situations, Bayesian estimates with a flat prior are identical to the ML estimates. Because Bayesian mapping provides a posterior distribution of QTL parameters, one automatically obtains the posterior variances and credibility intervals for the estimated parameters. Bayesian mapping is robust to the formulation of the model effects because all parameters are estimable if informative priors are used. This is clearly in contrast to ML, which frequently requires estimation of linear contrasts of unestimable model effects. One of the major hurdles of ML is finding the number of QTL. This involves a change in the dimensionality of the model. The recently developed reversible jump MCMC algorithm (Green 1995) allows the number of QTL to change in a convenient and objective way. This has revolutionized QTL mapping studies.

In Bayesian analysis, we treat parameters as unknown variables with a prior distribution. The purpose of Bayeisan analysis is to combine the prior distribution with the observed data to obtain a posterior distribution for the unknown paramaters. The summary statistic of the posterior distribution, for example, the mean, the mode, or the median, can be considered as Bayesian estimates. It should be noted that the prior distribution is not an actual distribution of the parameters. The parameters them-

selves are something fixed. It is our belief of the parameter values that is variable. Therefore, the prior distribution is actually the distribution of our subjective belief. Similarly, the posterior distribution of parameters is the updated distribution of our belief after incorporation of data.

8.3 Bayesian Mapping in Line Crosses

8.3.1 Probability Model

Let $\mathbf{y} = \{y_j\}$ for $j = 1, \ldots, N$ be a vector of phenotypic values for N individuals in the mapping population. Let q be the number of QTL, a_i and λ_i the effect and position, respectively, of the ith QTL for $i = 1, \ldots, q$. Define z_{ij} as the indicator variables for the genotype of the jth individual at the ith QTL. Assume that there are s markers available with known genome locations and define m_{kj} as the indicator variables for the jth individual at the kth marker. We now call $\mathbf{y}$ and $\mathbf{M} = \{m_{kj}\}$ the observables (data). The parameters of interest include q, $\mathbf{a} = \{a_i\}$, $\boldsymbol{\lambda} = \{\lambda_i\}$, μ, and σ_ε^2. The inheritance patterns (genotypes) of QTL, $\mathbf{Z} = \{z_{ij}\}$, are missing values. The parameters and missing values are grouped together and called unobservables, denoted by a vector $\boldsymbol{\theta}$. Using $p(x)$ and $p(x \mid y)$ as generic expressions for probability density and conditional probability density, respectively, where the actual form of the distribution does not depend on p but on its argument. The posterior distribution of $\boldsymbol{\theta}$ has the form of

$$p(\boldsymbol{\theta} \mid \mathbf{y}, \mathbf{M}) \propto p(\mathbf{y}, \mathbf{M} \mid \boldsymbol{\theta}) p(\boldsymbol{\theta}) \tag{8.9}$$

where $p(\mathbf{y}, \mathbf{M} \mid \boldsymbol{\theta}) = p(\mathbf{y} \mid \boldsymbol{\theta}) p(\mathbf{M} \mid \boldsymbol{\theta})$ is the likelihood and $p(\boldsymbol{\theta})$ is the prior probability density of the unobservables. Under the assumption of normal distribution for y, the first part of the likelihood is

$$p(\mathbf{y} \mid \boldsymbol{\theta}) = \frac{1}{\sqrt{2\pi\sigma_\varepsilon^2}} \exp\left[-\frac{1}{2\sigma_\varepsilon^2} \sum_{j=1}^{N} \left(y_j - \mu - \sum_{i=1}^{q} z_{ij} a_i\right)^2\right] \tag{8.10}$$

and the second part is

$$p(\mathbf{M} \mid \boldsymbol{\theta}) = p(\mathbf{M}, \mathbf{Z} \mid \boldsymbol{\lambda}, \gamma) / p(\mathbf{Z} \mid \boldsymbol{\lambda}) \tag{8.11}$$

where $\gamma = \{\gamma_k\}$ for $k = 1, \ldots, s$ is a *known* vector of positions for the s markers on the genome, $p(\mathbf{M}, \mathbf{Z} \mid \boldsymbol{\lambda}, \gamma)$ is the joint distribution of the genotypes of the s markers, and the q QTL and $p(\mathbf{Z} \mid \boldsymbol{\lambda})$ is the distribution of the genotypes of the q QTL. Both $p(\mathbf{M}, \mathbf{Z} \mid \boldsymbol{\lambda}, \gamma)$ and $p(\mathbf{Z} \mid \boldsymbol{\lambda})$ are determined by the positions of the QTL and markers.

Under the assumption of no interference between neighboring loci, they can be calculated using a Markov chain with two states. The transition probability between two loci is determined by the recombination fraction. Let $c_{i,i+1}$ be the recombination fraction between loci i and $i+1$. The transition probabilities between the two loci are $p(z_{i+1,j}=1 \mid z_{i,j}=1)=p(z_{i+1,j}=0 \mid z_{i,j}=0)=1-c_{i,i+1}$ and $p(z_{i+1,j}=1 \mid z_{i,j}=0)=p(z_{i+1,j}=0 \mid z_{i,j}=1)=c_{i,i+1}$. If locus i or $i+1$ is a marker, we simply replace $z_{i,j}$ or $z_{i+1,j}$ by $m_{i,j}$ or $m_{i+1,j}$. The recombination fraction relates to the map distance through the Haldane (1919) map function,

$$c_{i,i+1} = \frac{1}{2}[1-\exp(-2|\lambda_i - \lambda_{i+1}|)] \tag{8.12}$$

The joint prior probability density for the unobservables is further decomposed as

$$p(\boldsymbol{\theta}) = p(\mathbf{Z} \mid \boldsymbol{\lambda})p(q)p(\boldsymbol{\lambda})p(\mathbf{a})p(\mu)p(\sigma_\varepsilon^2) \tag{8.13}$$

where $p(\mathbf{Z} \mid \boldsymbol{\lambda}) = \prod_{j=1}^{N} p(\mathbf{Z}_j \mid \boldsymbol{\lambda})$ and $\mathbf{Z}_j = \{z_{ij}\}$ is the row vector of matrix $\mathbf{Z}$ that corresponds to the jth individual. As previously mentioned, $p(\mathbf{Z}_j \mid \boldsymbol{\lambda}) = p(z_{1j}, \ldots, z_{qj} \mid \boldsymbol{\lambda})$ is obtained using a heterogeneous Markov chain with two states. As an example, let us assume $q = 4$ and try to find the following joint probability,

$$\begin{aligned} & p(z_{1j}=0, z_{2j}=1, z_{3j}=0, z_{4j}=0 \mid \boldsymbol{\lambda}) \\ & \quad = p(z_{1j}=0)p(z_{2j}=1 \mid z_{1j}=0)p(z_{3j}=0 \mid z_{2j}=1)p(z_{4j}=0 \mid z_{3j}=0) \\ & \quad = \frac{1}{2}c_{12}c_{23}(1-c_{34}) \end{aligned}$$

where $p(z_{1j}=0)=\frac{1}{2}$ is the Mendelian prior. The prior for the QTL number can take a uniform distribution between 0 and $q_{\max}$. Sillanpää and Arjas (1998) used a Poisson prior. The prior for the locations of the QTL along the genome is $p(\boldsymbol{\lambda}) = \prod_{i=1}^{q} p(\lambda_i)$, where $p(\lambda_i)$ is uniform along the genome. Similarly, we use $p(\mathbf{a}) = \prod_{i=1}^{q} p(a_i)$, where $p(a_i)$ is either a uniform or a normal. A uniform prior is chosen for $p(\mu)$. Finally, a vague prior is taken for the residual variance, that is, $p(\sigma_\varepsilon^2) \propto 1/\sigma_\varepsilon^2$.

8.3.2 Markov Chain Monte Carlo

Given the complexity of the likelihood and the prior, the joint posterior probability density does not have a standard form. In addition, Bayesian inference should be made at the marginal level for each unobservable. Let us partition $\boldsymbol{\theta}$ into $\boldsymbol{\theta} = [\theta_i \boldsymbol{\theta}_{-i}]$, where θ_i is a single element of $\boldsymbol{\theta}$ and $\boldsymbol{\theta}_{-i}$ is a vector of the remaining elements. The marginal posterior distribution of θ_i is

$$p(\theta_i \mid \mathbf{y}, \mathbf{M}) \propto \iint p(\mathbf{y}, \mathbf{M} \mid \theta_i, \boldsymbol{\theta}_{-i}) p(\theta_i, \boldsymbol{\theta}_{-i})\, d\boldsymbol{\theta}_{-i} \tag{8.14}$$

Bayesian inference for θ_i should be made from the above marginal distribution. Unfortunately, this marginal distribution has no explicit expression. Numerical integration is often prohibited because of the high dimensionality of $\boldsymbol{\theta}_{-i}$. Therefore, we must use the MCMC algorithm to simulate a sample from the joint posterior distribution, $p(\boldsymbol{\theta} \mid \mathbf{y}, \mathbf{M})$. Using a realized sample, we can easily infer the marginal distribution of θ_i by simply looking at the empirical distribution of θ_i and ignoring the variation of $\boldsymbol{\theta}_{-i}$. With the MCMC algorithm, we do not directly generate the sample from $p(\boldsymbol{\theta} \mid \mathbf{y}, \mathbf{M})$; rather, we only generate realizations from the conditional posterior distribution for each unobservable, $p(\theta_i \mid \boldsymbol{\theta}_{-i}, \mathbf{y}, \mathbf{M})$. This conditional posterior distribution has an identical form to the joint posterior distribution except that, in the conditional distribution, $\boldsymbol{\theta}_{-i}$ is treated as constant and θ_i as a variable. Starting from an initial value for $\boldsymbol{\theta}$, denoted by $\boldsymbol{\theta}^{(0)} = [\theta_1^{(0)}, \ldots, \theta_r^{(0)}]$, where r is the total number of unobservables, we update one unobservable at a time with other unobservables fixed at their initial values. After all the unobservables have been updated, we complete one cycle of the Markov chain; the updated values are denoted by $\boldsymbol{\theta}^{(1)} = [\theta_1^{(1)}, \ldots, \theta_r^{(1)}]$. The chain will grow and eventually reach a stationary distribution. Let C be the length of the chain. Because there is one realization of $\boldsymbol{\theta}$ in each cycle of the chain, we will have a realized sample of $\boldsymbol{\theta}$ with sample size C, denoted by $\{\boldsymbol{\theta}^{(1)}, \ldots, \boldsymbol{\theta}^{(C)}\}$. Discarding data points of the first few thousand cycles (burn-in period) and thereafter saving one realization in every hundred cycles (to reduce the serial correlation between consecutive observations), we get a random sample of $\boldsymbol{\theta}$ drawn from $p(\boldsymbol{\theta} \mid \mathbf{y}, \mathbf{M})$.

Metropolis-Hastings Algorithm I now discuss how to sample θ_i from $p(\theta_i \mid \boldsymbol{\theta}_{-i}, \mathbf{y}, \mathbf{M})$. This conditional posterior distribution usually has a standard form, for example, normal. In this case, we can directly draw θ_i from the standard distribution. The method is called the Gibbs sampler (Geman and Geman 1984). If $p(\theta_i \mid \boldsymbol{\theta}_{-i}, \mathbf{y}, \mathbf{M})$ does not have a standard form, we will take a general acceptance-rejection approach, called the Metropolis-Hastings algorithm (Metropolis et al. 1953; Hastings 1970). Define $\theta_i^{(t-1)}$ as the values simulated at the $t-1$ cycle. We want to draw $\theta_i^{(t)}$ from the target distribution, $p(\theta_i \mid \boldsymbol{\theta}_{-i}, \mathbf{y}, \mathbf{M})$. Instead of directly drawing $\theta_i^{(t)}$ from this target distribution, the Metropolis-Hastings algorithm draws a candidate θ_i from a proposal density, $q(\theta_i^{(*)} \mid \theta_i^{(t-1)})$, which may be different from $p(\theta_i^{(*)} \mid \boldsymbol{\theta}_{-i}, \mathbf{y}, \mathbf{M})$ but has an easy form. We then use the Metropolis-Hastings rule to decide whether to accept $\theta_i^{(*)}$ or not. If $\theta_i^{(*)}$ is accepted, we let $\theta_i^{(t)} = \theta_i^{(*)}$; othewise, we simply let $\theta_i^{(t)} = \theta_i^{(t-1)}$. In either case, we will move to the next element. With the Metropolis-Hastings rule, we

accept $\theta_i^{(*)}$ with probability

$$\alpha = \min\left[\frac{p(\theta_i^{(*)} \mid \boldsymbol{\theta}_{-i}^{(t-1)}, \mathbf{y}, \mathbf{M}) q(\theta_i^{(t-1)} \mid \theta_i^{(*)})}{p(\theta_i^{(t-1)} \mid \boldsymbol{\theta}_{-i}^{(t-1)}, \mathbf{y}, \mathbf{M}) q(\theta_i^{(*)} \quad \theta_i^{(t-1)})}, 1\right] \tag{8.15}$$

The easiest form of the proposal density is $q(\theta_i^{(*)} \mid \theta_i^{(t-1)}) = w(\theta_i^{(*)} - \theta_i^{(t-1)})$ which characterizes the random walk chain. The candidate $\theta_i^{(*)}$ is drawn according to $\theta_i^{(*)} = \theta_i^{(t-1)} + \delta$, where δ is a noise random variable drawn from distribution $w(\delta)$. The actual form of $w(\delta)$ can be uniform or normal. In either case, $w(\theta_i^{(*)} - \theta_i^{(t-1)}) = w(\theta_i^{(t-1)} - \theta_i^{(*)})$, leading to

$$\alpha = \min\left[\frac{p(\theta_i^{(*)} \mid \boldsymbol{\theta}_{-i}^{(t-1)}, \mathbf{y}, \mathbf{M})}{p(\theta_i^{(t-1)} \mid \boldsymbol{\theta}_{-i}^{(t-1)}, \mathbf{y}, \mathbf{M})}, 1\right] \tag{8.16}$$

which is the Metropolis rule (Metropolis et al. 1953).

Reversible Jump MCMC The Gibbs sampler and Metropolis-Hastings algorithm described above can be used for updating all unobservables except q, the number of QTL. This is because parameter q is the dimension of the model and the Metropolis-Hastings algorithm in its original form only works when the dimensionality of the model is fixed. Green (1995) developed a reversible jump MCMC algorithm to accomplish the variable dimension problem. Sillanpää and Arjas (1998) applied this method to QTL mapping by drawing the number of QTL in BC mapping. Instead of drawing a proposed QTL number randomly and using the Metropolis-Hastings rule to accept the proposed QTL number, here we only consider one of two possibilities: add a new QTL to the model (with a probability p_a) or delete an existing QTL from the model (with probability $p_d = 1 - p_a$). Because q is also the dimension of the model, when q changes, the set of parameters will change accordingly.

Let us define the set of unobservables under the current model (with q QTL) by $\boldsymbol{\theta}^{(t-1)}$. If we propose to add a QTL, the new QTL number becomes $q^{(*)} = q + 1$. We should propose a new position (λ_{q+1}) and a new effect (a_{q+1}) corresponding to this new QTL. In addition, a genotype should be drawn for each individual corresponding to this new QTL $(z_{q+1,j})$. Define the additional unobservables after a new QTL has been added by $\mathbf{v} = [\lambda_{q+1}, a_{q+1}, \mathbf{Z}_{q+1}]$, where $\mathbf{Z}_{q+1} = \{z_{q+1,j}\}$ is a column vector corresponding to the genotypes of newly added QTL. Therefore, the proposed set of parameters becomes $\boldsymbol{\theta}^{(*)} = [\boldsymbol{\theta}^{(t-1)}, \mathbf{v}]$. We now accept the new QTL with probability

$$\alpha = \min\left[\frac{p(\boldsymbol{\theta}^{(*)} \mid \mathbf{y}, \mathbf{M}) q(\boldsymbol{\theta}^{(t-1)} \mid \boldsymbol{\theta}^{(*)})}{p(\boldsymbol{\theta}^{(t-1)} \mid \mathbf{y}, \mathbf{M}) q(\boldsymbol{\theta}^{(*)} \mid \boldsymbol{\theta}^{(t-1)}) p(\mathbf{v})}, 1\right] \tag{8.17}$$

where $p(\mathbf{v}) = p(\lambda_{q+1})p(a_{q+1})p(\mathbf{Z}_{q+1})$, $p(\lambda_{q+1})$ is uniform along the genome, $p(a_{q+1})$ is uniform or normal highly concentrated around 0, and $p(\mathbf{Z}_{q+1})$ is the Mendelian prior for the genotype of the new QTL. The proposal probability of adding a QTL is predefined by p_a, and thus, $q(\boldsymbol{\theta}^{(*)} \mid \boldsymbol{\theta}^{(t-1)}) = p_a$. The reverse process is to delete the $(q+1)$th QTL from model $\boldsymbol{\theta}^{(*)}$ to obtain model $\boldsymbol{\theta}^{(t-1)}$. Because each one of the $q+1$ QTL has an equal chance to be deleted, the probability that the deletion happens to be the $(q+1)$th QTL is $1/(q+1)$. Therefore, $q(\boldsymbol{\theta}^{(t-1)} \mid \boldsymbol{\theta}^{(*)}) = p_d/(q+1)$. If $\boldsymbol{\theta}^{(*)}$ is accepted, $\boldsymbol{\theta}^{(t)} = \boldsymbol{\theta}^{(*)}$, the new QTL is added to the model and all the unobservables corresponding to the new QTL are accepted simultaneously. If $\boldsymbol{\theta}^{(*)}$ is rejected, $\boldsymbol{\theta}^{(t)} = \boldsymbol{\theta}^{(t-1)}$, and the model stays the same. However, all the unobservables are updated using the regular Metropolis-Hastings algorithm as described before.

Deleting a QTL simply takes the reverse process of adding a QTL. Again, define $\boldsymbol{\theta}^{(t-1)}$ as the set of unobservables under the current model with q QTL. We now define $\boldsymbol{\theta}^{(*)}$ as the set of unobservables after one QTL has been deleted from the model, with $q^{(*)} = q - 1$ QTL. The relationship between $\boldsymbol{\theta}^{(t-1)}$ and $\boldsymbol{\theta}^{(*)}$ is $\boldsymbol{\theta}^{(t-1)} = [\boldsymbol{\theta}^{(*)}, \mathbf{v}]$, where $\mathbf{v} = [\lambda_q, a_q, \mathbf{Z}_q]$. The proposal probability is $q(\boldsymbol{\theta}^{(*)} \mid \boldsymbol{\theta}^{(t-1)}) = p_d/q$ and the density of the reverse process is $q(\boldsymbol{\theta}^{(t-1)} \mid \boldsymbol{\theta}^{(*)}) = p_a$. The probability of accepting the deletion is

$$\alpha = \min\left[\frac{p(\boldsymbol{\theta}^{(*)} \mid \mathbf{y}, \mathbf{M})q(\boldsymbol{\theta}^{(t-1)} \mid \boldsymbol{\theta}^{(*)})p(\mathbf{v})}{p(\boldsymbol{\theta}^{(t-1)} \mid \mathbf{y}, \mathbf{M})q(\boldsymbol{\theta}^{(*)} \mid \boldsymbol{\theta}^{(t-1)})}, 1\right] \tag{8.18}$$

Similar to the process of adding QTL, if the proposal is rejected, the model dimension will stay the same, but each unobservable in the existing model will be updated according to the regular Metropolis-Hastings rule.

8.3.3 Post–Bayesian Analysis

Data sampled from the posterior distribution contain all the information we need to infer the statistical properties of the parameters, so, the MCMC algorithm serves as an experiment to generate data. After the experiment, we need to summarize the data and draw conclusions. In fact, the statistical properties of parameters are "directly observed" from the data rather than inferred as in usual data analyses. This is because the sampled data points from the posterior distribution are made directly on the parameters.

Empirical Distribution The most informative summary from the posterior sample is the frequency table for each parameter of interest. The table may be converted into a histogram, which is a visual representation of the posterior density. The posterior mean, posterior variance, and credibility interval are also easily obtained from the

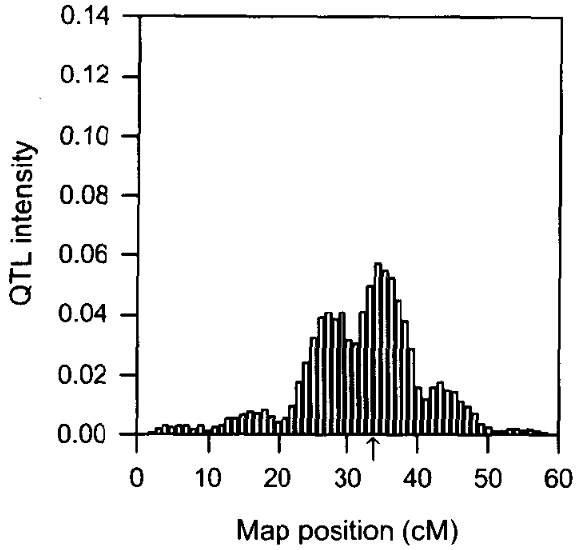


Figure 8.1
Posterior density of QTL location obtained from a simulation experiment. The true location of the QTL is indicated by the arrow.

posterior sample. If a proper and continuous prior is chosen for each parameter, we expect that the posterior distribution is asymptotically normal. Therefore, a severe deviation from normality indicates insufficient sample size. The summary statistics of the posterior distribution are useful for QTL parameters when a single QTL is fitted to the model. The most important parameter of interest is the location of the QTL in the genome. The marginal posterior distribution of the QTL position can be depicted via plotting the number (frequency) of QTL occurrence in a short interval against the genome location of the interval. The regions frequently hit by the QTL are candidate locations of the QTL. The uncertainty of each candidate region is reflected by the width of the peak in the posterior density (see the posterior distribution of QTL location depicted in figure 8.1).

For multiple QTL, we use the reversible jump MCMC for the change of model dimension. As the number of QTL frequently changes, most QTL have lost their identities. For instance, the first QTL in one observation may not be the first QTL in another observation if new QTL have been added. When the QTL lose their identities, the posterior distributions of the corresponding QTL effects also lose their meanings. Although the posterior distributions of q, μ, and σ_ε^2 are still meaningful in the multiple QTL model, we must seek alternative representations of the summary statistics for other QTL parameters.

QTL Intensity and Profile of QTL Effect As mentioned earlier, the posterior density of the location of a QTL is estimated by the proportion of the number of hits by the

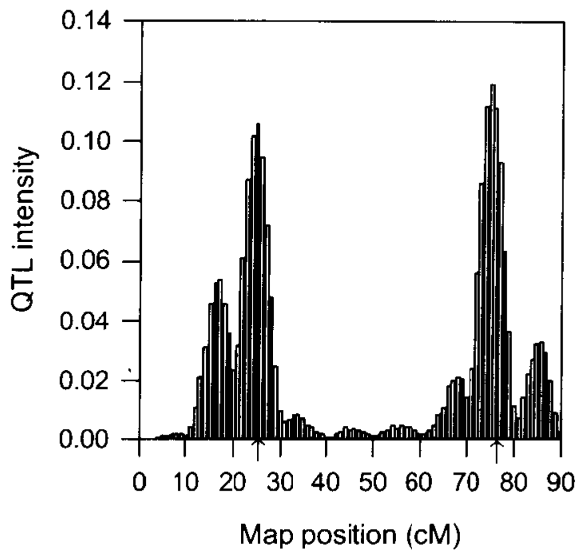


Figure 8.2
Profile of QTL intensity obtained from a simulation experiment. The true locations of the two simulated QTL are indicated by the arrows.

QTL to a short interval surrounding that location. When a QTL loses its identity, we are unable to keep track of the hits by individual QTL; rather, we can only keep record of the total number of hits to a particular interval. Multiple hits to a short interval may be due to different hits of the same QTL from different observations or due to multiple hits by different QTL from the same observation. As a consequence, we completely ignore the origins of the hits and record the total number of hits by QTL along the whole genome. We then divide the whole genome into many equidistant short intervals, say 1 cM, and count the number of hits to each short interval. The proportion of the hits to each interval, $P(t)$, is plotted against t, the genome location of the interval. In contrast to a single QTL model, the curve is no longer called the posterior density of QTL location; rather, it is called the QTL intensity profile (see the QTL intensity profile in figure 8.2). Therefore, the posterior density of QTL location and QTL intensity are used interchangably, only under a single QTL model.

Similarly, when the identity of a QTL is lost, the effects associated with individual QTL also lose their meanings. Corresponding to the QTL intensity profile, we calculate the average effect for each of the short intervals of the genome (sum of the QTL effects of multiple hits divided by the number of hits) and form a profile for the QTL effect, $E(t)$. For the candidate regions of QTL (regions repeatedly hit by QTL), we can visualize the average effect of QTL in those regions. One should be cautious that sometimes the profile of the QTL effect can be misleading. We have noticed that regions rarely hit by QTL can sometimes show a large average effect (see the QTL

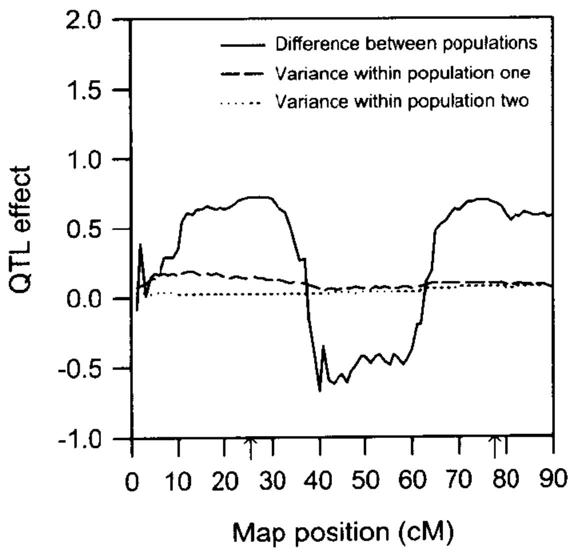


Figure 8.3
Profiles of QTL effects obtained from a simulation experiment. The true locations of the two simulated QTL are indicated by the arrows.

effect profiles in figure 8.3). The effect profile is only meaningful for regions with high QTL intensity.

A weighted QTL intensity was recently proposed (Xu and Yi 2000) in which the product of the QTL intensity and QTL effect, denoted by $W(t) = P(t)E(t)$, is plotted against the chromosomal position. This weighted QTL intensity will eliminate the peaks in the regions rarely hit by QTL, even if the average effect in the regions is large. For other designs in which dominance effect can be examined, there are two weighted QTL intensities: (1) the weighted additive intensity, $W_a(t) = P(t)A(t)$, where $A(t)$ is the average additive effect at location t; and (2) the weighted dominance intensity, $W_d(t) = P(t)D(t)$, where $D(t)$ is the average dominance effect at location t. By looking at the weighted QTL intensities, we can immediately tell the sources of genetic variances (additive and dominance) of the QTL in a particular region of the genome (see figure 8.4 for the weighted QTL intensity profiles).

8.4 Bayesian Mapping in Pedigrees

Pedigrees are used for QTL mapping in outbred populations, such as humans, where the development and crossing of inbred lines is not feasible. Most domesticated animal and plant populations are not inbred. They are usually bred in pedigrees with complicated mating designs, such as a diallel cross design. Even if inbred lines are used, different crosses may be connected by one or two common ancestors. QTL

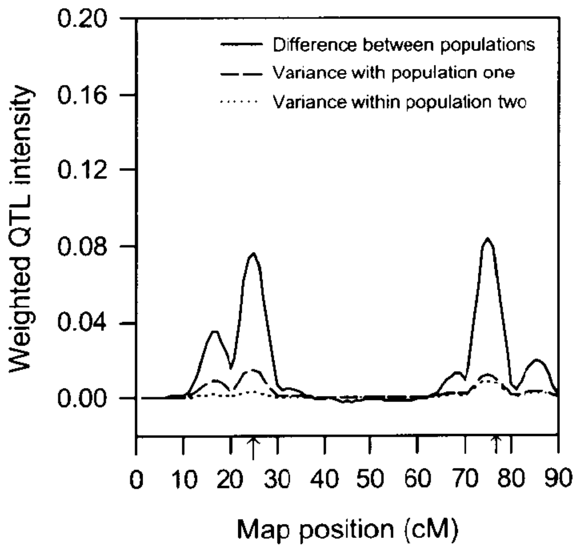


Figure 8.4
Weighted QTL intensity profiles obtained from a simulation experiment. The true locations of the two simulated QTL are indicated by the arrows.

mapping in such a population with irregular mating systems is complicated enough to consider use of the pedigree analysis approach. The Bayesian method via the MCMC algorithm is the ideal tool for pedigree analysis.

8.4.1 Mixed Model

Consider a hybrid population contributed by several genetically distinguishable source populations. The hybrid population has gone through one or more generations of random mating. All individuals in the mapping population have a complete record of pedigree traced back to the ancestors in the source populations. In pedigree analysis, it is convenient to deal with alleles (haploid) rather than genotypes (diploid). Therefore, the genetic parameters are defined exclusively in terms of allelic rather than genotypic values. We consider only a single locus in the description of the mixed model theory, and multiple loci will be discussed in a later section. Let S be the number of source populations, and define the expectation and variance of the allelic values for population k by b_k and σ_k^2, respectively. For diploid organisms, both the mean and variance of the additive genetic values take twice the values of their allelic counterparts. Assume that all the source populations are of equal size. The total allelic variance of the combined population in the current generation is

$$\sigma_A^2 = \frac{1}{S}\left[\sum_{k=1}^{S}\sigma_k^2 + \sum_{k=1}^{S}(b_k - \bar{b})^2\right] \tag{8.19}$$

where $\bar{b} = \frac{1}{S}\sum_{k=1}^{S} b_k$. This partitioning of the total genetic variance indicates that the locus under study contributes to the trait variation if at least one of the components is not zero.

Let $\frac{1}{2}n_k$ be the numbers of contributing parents from population k so that the number of founder alleles from this population is n_k. Each of the n_k allelic values is assumed to be randomly sampled from population k with a $N \sim (b_k, \sigma_k^2)$ distribution. The mean value of the n_k alleles, b_k, is a fixed effect. Therefore, the model is called a mixed model. Assume that a parent from one population has an equal chance to mate with any parent from the other population. The mating of the F_1's are completely random, so that the alleles of the original populations are well integrated into the hybrid population. We can take F_2 as our mapping population, but including advanced generations can be more efficient because the alleles from different populations are well mixed. In this case, we can estimate each variance component of σ_A^2. Unfortunately, such a mating design produces complex pedigrees that prevent the use of a simple method for estimation. Assume that there are N individuals in the mapping population. We denote the effects of the paternal and maternal alleles of individual j by v_j^p and v_j^m, respectively, for $j = 1, \ldots, N$. The phenotypic value of individual j can be described by the following linear model:

$$y_j = \mu + v_j^p + v_j^m + \varepsilon_j \tag{8.20}$$

where μ is the population mean (fixed effect) and ε_j is the residual error with a $N(0, \sigma_\varepsilon^2)$ distribution. Using the notation of Fernando and Grossman (1989), we define v_p^p and v_p^m as the paternal and maternal alleles for the father of j so that

$$v_j^p = z_j^p v_p^p + (1 - z_j^p) v_p^m \tag{8.21}$$

where z_j^p indicates the allelic inheritance of the paternal allele of the father. Similarly, define v_m^p and v_m^m as the paternal and maternal alleles of the mother and

$$v_j^m = z_j^m v_m^p + (1 - z_j^m) v_m^m \tag{8.22}$$

where z_j^m indicates the allelic inheritance of the paternal allele of the mother. The above model can be rewritten as

$$y_j = \mu + z_j^p v_p^p + (1 - z_j^p) v_p^m + z_j^m v_m^p + (1 - z_j^m) v_m^m + \varepsilon_j \tag{8.23}$$

We have now expressed the allelic values of the current generation as a linear function of allelic values in the parental generation. The parental alleles can be further expressed as a linear function of the allelic values of their parents. With such a recurrent process, each allele can be traced back to its origin in the S founder popu-

lations. Let $n = \sum_{k=1}^{S} n_k$ be the total number of founder alleles from all the source populations and define $\mathbf{a}$ as an $n \times 1$ vector containing the values of all the founder alleles. Assume that the founder alleles in $\mathbf{a}$ are ordered according to their source populations, that is, the first n_1 elements of $\mathbf{a}$ store the founder alleles from the first source population. If a_i, for $i = 1, \ldots, n$, comes from source population k, we assume $a_i \sim N(b_k, \sigma_k^2)$. The linear model in matrix notation is

$$\mathbf{y} = \mathbf{1}\mu + (\mathbf{Z}^p + \mathbf{Z}^m)\mathbf{a} + \varepsilon \tag{8.24}$$

where $\mathbf{Z}^p$ and $\mathbf{Z}^m$ are $N \times n$ indicator matrices connecting the paternal and maternal alleles of all individuals to the founder alleles. Let $\mathbf{b} = [b_1, \ldots, b_S]'$ and $\mathbf{W} = \text{diag}\{\mathbf{1}_{n_1}, \ldots, \mathbf{1}_{n_S}\}$. We can describe $\mathbf{a}$ by a linear function, $\mathbf{a} = \mathbf{W}\mathbf{b} + \mathbf{u}$, where $\mathbf{u} = \{u_i\}$ is an $n \times 1$ vector of variables corresponding to $\mathbf{a}$ but with $u_i \sim N(0, \sigma_k^2)$ being assumed if u_i comes from the kth source population. Substituting $\mathbf{a}$ in equation (8.24) by the above linear combination, we get

$$\mathbf{y} = \mathbf{1}\mu + (\mathbf{Z}^p + \mathbf{Z}^m)\mathbf{W}\mathbf{b} + (\mathbf{Z}^p + \mathbf{Z}^m)\mathbf{u} + \varepsilon \tag{8.25}$$

Let $\mathbf{X} = (\mathbf{Z}^p + \mathbf{Z}^m)\mathbf{W}$ and $\mathbf{Z} = \mathbf{Z}^p + \mathbf{Z}^m$. The above model is then expressed as a typical mixed model:

$$\mathbf{y} = \mathbf{1}\mu + \mathbf{X}\mathbf{b} + \mathbf{Z}\mathbf{u} + \varepsilon \tag{8.26}$$

where $\mathbf{b}$ is the vector of fixed effects and $\mathbf{u}$ is the vector of random effects with zero expectation and a variance matrix of $\mathbf{G} = \text{diag}\{\mathbf{I}\sigma_1^2, \ldots, \mathbf{I}\sigma_S^2\}$. This model is different from the usual mixed model in that the design matrices, $\mathbf{X}$ and $\mathbf{Z}$, are unknown because the actual allelic inheritance of the QTL is not observable. What we can observe is the allelic inheritance of markers in the neighborhood of the QTL. These markers can be used to infer $\mathbf{Z}$ and thus $\mathbf{X}$.

8.4.2 Probability Model

Denote the allelic inheritance patterns of markers by $\mathbf{M}$. The class of observables includes $\mathbf{y}$ and $\mathbf{M}$. The class of unobservables includes the parameters of interest $\{\mu, \mathbf{b}, \sigma_1^2, \ldots, \sigma_S^2, \sigma_\varepsilon^2, \lambda\}$ and the missing factors $\{\mathbf{Z}, \mathbf{u}\}$, where λ is the position of the QTL. Again, we use a single QTL model to demonstrate the procedure and discuss the multiple QTL model in a later section. Define the collection of unobservables as $\boldsymbol{\theta} = \{\mu, \mathbf{b}, \sigma_1^2, \ldots, \sigma_S^2, \lambda, \sigma_\varepsilon^2, \mathbf{Z}, \mathbf{u}\}$. The joint posterior probability density of $\boldsymbol{\theta}$ is

$$p(\boldsymbol{\theta} \mid \mathbf{y}, \mathbf{M}) \propto p(\mathbf{y}, \mathbf{M} \mid \boldsymbol{\theta})p(\boldsymbol{\theta}) \tag{8.27}$$

The joint prior probability is defined as

$$p(\boldsymbol{\theta}) = p(\mu)p(\mathbf{b})p(\sigma_1^2)\dots p(\sigma_S^2)p(\lambda)p(\sigma_\varepsilon^2)p(\mathbf{Z})p(\mathbf{u}\,|\,\sigma_1^2,\dots,\sigma_S^2) \tag{8.28}$$

We take uniform priors for all the unobservables except $\mathbf{Z}$ and $\mathbf{u}$ in which we take

$$p(\mathbf{u}\,|\,\sigma_1^2,\dots,\sigma_2^2) \propto \frac{1}{(\sigma_1^2)^{n_1/2}\dots(\sigma_S^2)^{n_S/2}} \exp\left\{-\frac{1}{2}\mathbf{u}'\mathbf{G}^{-1}\mathbf{u}\right\} \tag{8.29}$$

The Mendelian prior is taken for $\mathbf{Z}$ which will be discussed in the next section. The likelihood is

$$p(\mathbf{y},\mathbf{M}\,|\,\boldsymbol{\theta}) = p(\mathbf{y}\,|\,\mu,\mathbf{b},\mathbf{Z},\mathbf{u},\sigma_\varepsilon^2)p(\mathbf{M}\,|\,\mathbf{Z},\lambda) \tag{8.30}$$

Given $\mathbf{X}$ and $\mathbf{Z}$, our linear model is a standard mixed model. Bayesian inference of variance components under the standard mixed model has been extensively studied (e.g., Wang et al. 1993; Clayton 1999). Herein, we only describe methods of generating $\mathbf{Z}$, evaluating the likelihood and simulating $\mathbf{b}$ and $\mathbf{u}$, given other unobservables fixed at values previously simulated.

8.4.3 Evaluating the Likelihood Function

The likelihood has been factorized into that of the phenotypic values of the trait and that of marker genotype, that is, $p(\mathbf{y},\mathbf{M}\,|\,\boldsymbol{\theta}) = p(\mathbf{y}\,|\,\boldsymbol{\theta})p(\mathbf{M}\,|\,\boldsymbol{\theta})$. Conditional on their genotypic values, the phenotypic values of any two individuals are independent. This leads to a convenient way to evaluate the likelihood

$$p(\mathbf{y}\,|\,\mu,\mathbf{b},\mathbf{Z},\mathbf{u},\sigma_\varepsilon^2) = \prod_{j=1}^{N} p(y_j\,|\,\mu,v_j^p,v_j^m,\sigma_\varepsilon^2) \tag{8.31}$$

The fact that v_j^p and v_j^m can be expressed as functions of the allelic values of the parents of individual j allows dynamic programming to be used for evaluation of the likelihood function. This algorithm requires individuals to be entered into the pedigree in the chronological order of their birth so that the likelihoods of parents are always evaluated before their children (van Arendonk et al. 1994).

Starting from given values of $\mathbf{b}$ and $\mathbf{u}$, we obtain $\mathbf{a} = \mathbf{Wb} + \mathbf{u}$, which contains n allelic values of the original founders in the S source populations. Let $a(i)$ be the value of the ith founder allele for $i = 1,\dots,n$. Instead of using the conventional notation of a_i for the ith element of vector $\mathbf{a}$, here we use a pseudo-code notation $a(i)$. Recall that each allele in the mapping population can eventually be traced back to one (and only one) of the founder alleles. Define i_j^p and i_j^m as the identifiers of the paternal and maternal alleles of individual j in the founder alleles. For example, if the paternal and maternal alleles of individual j originated from the fourth

and eighth founder alleles, respectively, then $i_j^p = 4$, $i_j^m = 8$, $v_j^p = a(i_j^p) = a(4)$ and $v_j^m = a(i_j^m) = a(8)$. This leads to a pseudo-code expression of the linear model for individual j,

$$y_j = \mu + a(i_j^p) - a(i_j^m) + \varepsilon_j \tag{8.32}$$

The identifiers, i_j^p and i_j^m, can be easily found using a dynamic programming approach as shown below. Because individuals must be evaluated in chronological order, parents must be evaluated before their progeny. If individual j is a founder and it is the f founder, then $i_j^p = 2f - 1$ and $i_j^m = 2f$ for $f = 1, \ldots, \frac{1}{2}n$. If j is not a founder, we know that the parents of j must have been evaluated. Define i_p^p and i_p^m as the identifiers for the paternal and maternal alleles of j's father and i_m^p and i_m^m as the identifiers for the paternal and maternal alleles of j's mother. Given the allelic identifiers of the parents, the identifiers of j are calculated as

$$i_j^p = z_j^p i_p^p + (1 - z_j^p) i_p^m \tag{8.33}$$

and

$$i_j^m = z_j^m i_m^p + (1 - z_j^m) i_m^m \tag{8.34}$$

where z_j^p and z_j^m are the parental allelic inheritance indicators for individuial j, as defined earlier.

This dynamic programming approach to evaluating the likelihood is extremely simple. It only requires four variables for each individual, $\{i_j^p\}$, $\{i_j^m\}$, $\{z_j^p\}$, and $\{z_j^m\}$. Yet the algorithm is so powerful that no restrictions are made in terms of the complexity of the pedigrees. If $i_j^p = i_j^m$, then j is inbred at the locus of interest. If any one of the following, $i_j^p = i_k^p$, $i_j^p = i_k^m$, $i_j^m = i_k^p$, or $i_j^m = i_k^m$, is true, individuals j and k are genetically related. Therefore, the algorithm can be applied to QTL mapping in arbitrarily complicated mating systems, including selfing.

The likelihood of the markers $p(\mathbf{M} \mid \boldsymbol{\theta})$ is evaluated by taking the product of individual-wise likelihood. For individual j, the marker likelihood is derived as follows. Define the QTL genotypes of the father and mother by $Q_p^p Q_p^m$ and $Q_m^p Q_m^m$, respectively. Individual j can take one of the four possible ordered genotypes $\{Q_p^p Q_m^p, Q_p^p Q_m^m, Q_p^m Q_m^p, Q_p^m Q_m^m\}$. Define $\mathbf{U}_j = [U_{1j}\ U_{2j}\ U_{3j}\ U_{4j}]$ as a vector of indicator variables where $U_{kj} = 1$ and $U_{k'j} = 0$ for $k' \neq k$ if the progeny takes the kth ordered genotype. Define the jth row of $\mathbf{Z}$ by $\mathbf{Z}_j = [z_j^p, 1 - z_j^p, z_j^m, 1 - z_j^m]'$. We can see that $\mathbf{Z}_j$ and $\mathbf{U}_j$ have a linear relationship, that is, $z_j^p = U_{1j} + U_{2j}$ and $z_j^m = U_{1j} + U_{3j}$. Similar to $\mathbf{U}_j$, we denote the genotype indicator vectors for the left and right markers flanking the QTL by $\mathbf{M}_j^L$ and $\mathbf{M}_j^R$, respectively. The likelihood of markers condi-

tional on $\mathbf{Z}_j$, and thus $\mathbf{U}_j$, can be expressed by

$$p(M_{sj}^L, M_{tj}^R \mid U_{kj}, \lambda) = p(M_{sj}^L \mid U_{kj}, \lambda) p(M_{tj}^R \mid U_{kj}, \lambda) \tag{8.35}$$

where $p(M_{sj}^L \mid U_{kj}, \lambda)$ or $p(M_{tj}^R \mid U_{kj}, \lambda)$ is found from the following transition matrix:

$$\mathbf{P}_{MU} = \begin{bmatrix} (1-c_{MU})^2 & (1-c_{MU})c_{MU} & (1-c_{MU})c_{MU} & c_{MU}^2 \\ (1-c_{MU})c_{MU} & (1-c_{MU})^2 & c_{MU}^2 & (1-c_{MU})c_{MU} \\ (1-c_{MU})c_{MU} & c_{MU}^2 & (1-c_{MU})^2 & (1-c_{MU})c_{MU} \\ c_{MU}^2 & (1-c_{MU})c_{MU} & (1-c_{MU})c_{MU} & (1-c_{MU})^2 \end{bmatrix}$$

where c_{MU} is the recombination fraction between the QTL and the left or right marker. It is calculated from λ using the Haldane (1919) map function.

Because only two flanking markers are used to calculate the posterior probability of U_{kj}, the approach is called interval mapping (Lander and Botstein 1989). If a marker, for example, M_{sj}^L, is not fully informative, we generate a realization of M_{sj}^L based on markers flanking M_{sj}^L. Alternatively, we can take the multipoint method using all markers on the lefthand side of the QTL in place of M_{sj}^L.

8.4.4 Sampling the Unobservables

Given the relationships, $z_j^p = U_{1j} + U_{2j}$ and $z_j^m = U_{1j} + U_{3j}$, the problem of sampling z_j^p and z_j^m has become that of sampling $\mathbf{U}_j$. Here we take a Gibbs sampling approach and simulate $\mathbf{U}_j$ directly from its posterior distribution. The posterior distribution of U_{kj} is

$$p(U_{kj} \mid y_j, M_{sj}^L, M_{tj}^R, \theta) = \frac{p(y_j \mid \mu, U_{kj}, v_j^p, v_j^m, \sigma_\varepsilon^2) p(M_{sj}^L, M_{tj}^R \mid U_{kj}, \lambda) p(U_{kj})}{\sum_{k=1}^4 p(y_j \mid \mu, U_{kj}, v_j^p, v_j^m, \sigma_\varepsilon^2) p(M_{sj}^L, M_{tj}^R \mid U_{kj}, \lambda) p(U_{kj})} \tag{8.36}$$

for $k, s, t = 1, \ldots, 4$ where the Mendelian prior $p(U_{kj}) = 1/4$ for $k = 1, \ldots, 4$ is taken.

Sampling μ, σ_ε^2, and λ has been described in the section of line crosses and will not be further discussed here. We now describe the sampling process for other parameters. The vector of fixed effects, $\mathbf{b} = \{b_k\}$, can be sampled simultaneously (block-wise) or separately using a random walk Metropolis algorithm. As in the usual mixed model analysis, we put a constraint on the fixed effects to obtain a meaningful estimate of $\mathbf{b}$ by letting $b_1 = 0$. The random effects $\mathbf{u}$ are sampled in the same way as $\mathbf{b}$ except that no constraint is needed here. Because the posterior distributions of $\mathbf{b}$ and $\mathbf{u}$ are normals, a Gibbs sampler may also be used. The allelic variance of each source population, σ_k^2, is sampled via the Metropolis-Hastings algorithm. Although a Gibbs sampler

may be used, the posterior distribution may be hard to derive. For a multiple QTL model, we need to sample the number of QTL using the reversible jump MCMC as described earlier.

8.4.5 Dominance and Epistatic Effects

Recall that the genetic value of individual j has been defined as $v_j^p + v_j^m$. The sum is also called the additive genetic value or breeding value. In fact, the genotypic value, denoted by g_j, may be different from the breeding value due to interaction between the two alleles. This deviation is called the dominance effect, denoted by $\delta_j = g_j - (v_j^p + v_j^m)$. Therefore, the dominance genetic model is

$$y_j = \mu + v_j^p + v_j^m + \delta_j + \varepsilon_j \tag{8.37}$$

Similar to the allelic value that can be traced back to one of the founder alleles, the dominance effect can also be traced back to the interaction effect between two founder alleles. For a total of n founder alleles, there are $n(n + 1)/2$ possible interactions, including the interaction of an allele with its own copy. Let $\mathbf{d} = \{d_{ij}\}$ for $i, j = 1, \ldots, n$ be an $n \times n$ symmetric matrix storing all the interaction effects of the founder alleles. While two founder alleles are transmitted to a progeny, the interaction effect between the two is also passed to the progeny. Therefore, the pseudo-code expression of the dominance model is

$$y_j = \mu + a(i_j^p) + a(i_j^m) + d(i_j^p, i_j^m) + \varepsilon_j \tag{8.38}$$

The dominance effect of the founder alleles, d_{ij}, has a distribution depending on the origins of the two alleles. If allele i comes from the kth source population and allele j comes from the lth source population, then $d_{ij} \sim N(\mu_{kl}, \gamma_{kl}^2)$ is assumed, where μ_{kl} is the mean effect of dominance and γ_{kl}^2 is the dominance variance. The number of parameters grows quickly as the number of source populations increases. Therefore, to examine dominance effects, one should choose to use fewer source populations.

If a trait is controlled by multiple QTL, an allele from one QTL may interact with an allele from another QTL. This interaction effect is called the additive-by-additive epistatic effect, or simply epistatic effect. Higher order interaction between alleles from different loci may also occur. For example, if two alleles from locus one interact with one allele from locus two, the interaction effect is called the dominance-by-additive epistatic effect. By the same token, the dominance-by-dominance effect results from the interaction of four alleles. The additive-by-additive effect is easy to handle and is usually more important than the higher order epistatic effects. Therefore, only the additive-by-additive effect is discussed. For simplicity, let us consider a model with two loci only. Define $\mathbf{A} = \{a_{ij}\}$ as an $n \times 2$ matrix, each column containing values of

the n founder alleles for each locus. Also define $\mathbf{D} = \{d_{ijk}\}$ as a $2 \times n \times n$ (three-dimension) matrix with $\{d_{1jk}\}$ and $\{d_{2jk}\}$ denoting the dominance effects among the founder alleles of loci one and two, respectively. Further, define $\mathbf{H} = \{h_{ij}\}$ as an $n \times n$ matrix with h_{ij} denoting the interaction (epistatic) effect between the ith founder allele from locus one and the jth founder alleles from locus two. Similar to i_j^p and i_j^m, let us define the allele identifiers for the second locus as k_j^p and k_j^m. The epistatic model is

$$y_j = \mu + a(1, i_j^p) + a(1, i_j^m) + d(1, i_j^p, i_j^m) + a(2, k_j^p) + a(2, k_j^m) + d(2, k_j^p, k_j^m) \\ + h(i_j^p, k_j^p) + h(i_j^p, k_j^m) + h(i_j^m, k_j^p) + h(i_j^m, k_j^m) + \varepsilon_j \tag{8.39}$$

Similar to dominance effects, each epistatic effect is assumed to follow a normal distribution with mean and variance depending on the origins of the two alleles in the source populations. The number of parameters can be large for a large number of source populations. Therefore, epistatic effects can be examined with sufficient accuracy only if the sample size is large.

8.5 Discussion

Because the phenotype of a quatitative trait is determined jointly by the effect of genes and an environmental error, the behaviors of individual genes cannot be examined separately using the traditional quantitative genetics techniques; rather, they must be studied collectively using indirect information such as phenotype of the trait in question, phenotypic values of correlated traits and phenotypic measurements of genetically related individuals. Linkage analysis of quantitative traits uses more direct information such as molecular markers and candidate genes, in addition to phenotype and pedigree information. Markers are located along the chrosmosomes where the QTL may reside. As a result, segregation patterns of markers partially reflect those of the QTL. If a marker actually overlaps with a QTL, the segregation of the QTL is observed through the marker. Furthermore, segregations of molecular markers are usually not affected by the change of environmental factors, so the information that comes from markers is more direct than the phenotypic information. As the development of molecular technology, marker maps can be made arbitrarily dense so that all genes are actually observed. Genetic improvement of organisms can be made more efficient if genes can be directly manipulated.

The mixed model approach implemented via the MCMC algorithm provides a unified QTL mapping algorithm. It can analyze data collected from arbitrarily complicated mating designs, including simple line crossing experiments and random mating populations. If all the source populations are inbred lines, we know a priori

that $\sigma_k^2 = 0$ for $k = 1, \ldots, S$, so that the total genetic variance is contributed only by the bewteen-line difference, the second term of equation (8.19). In this case, the model is simply a fixed model, as commonly seen in the QTL mapping literature. The simple F_2 or BC design of QTL mapping is a special case when $S = 2$. On the other hand, if the mapping population is a random sample from a single large homogeneous (unstructured) population ($S = 1$), the bewteen-line variance is a prori set to zero and only the within-line variance, the first term of equation (8.19) is retained in the total genetic variance. The method then becomes a random model approach that is commonly used in QTL mapping for outbred populations. With a single statement to turn on/off the option of fixed/random, the mixed model approach can handle a mating design with any level of complexity. This unified QTL mapping program can potentially replace most mapping programs currently available, and thus eliminate unnecessary comparisons of the major mapping programs commonly seen in the reports of QTL experiments.

The major hurdle in implementing the unified Bayesian mapping algorithm is the high demand for computer power. To analyze a single data set with a few thousand individuals, the analysis can take a few days, whereas the simple least square method or maximum likelihood method, if available, may take only a few minutes. The other limitation is the high demand for memory storage when the number of founders is large. Although this is not a big problem for the additive model, the problem can be very serious for dominance and epistatic models with a large number of QTL. For each QTL, we need to save $n(n+1)/2$ dominance effects in the founders. For every pair of QTL, we need to save n^2 epistatic effects. The amount of memory storage grows quickly as the number of QTL increases. Therefore, for the epistatic model, one must restrict the founders to a reasonable number, say $n \leq 100$. When n is large, we may consider using a multiplicative dominance and epistatic model. This model assumes that the interaction (dominance or epistatic) effect between two alleles is proportional to the product of the two allelic (additive) values. The proportion is $\sigma_I/(\sigma_k\sigma_l)$, where σ_I is the standard deviation of the interaction effect, and σ_k and σ_l are the standard deviations of alleles k and l, respectively. With the multiplicative interaction effect model, we only need to save the allelic values of founders. As a consequence, one can easily incorporate higher order interactions, for example, dominance-by-dominance. We can even consider interactions of alleles among multiple loci.

Marker genotypes have been treated as data (observables) in this study. In situations where an individual does not have a complete array of genotypes, special algorithms are required to handle missing genotypes. If the individual with missing genotypes is a progeny (no children), one can simply skip the missing marker and use

markers nearby to infer the allelic inheritance of a QTL. If the individual is a founder or a parent, one needs to recover the missing genotypes using available marker genotypes of all its relatives. The method of descent graphs is particularly designed for this purpose (Sobel and Lange 1996). Implementation of descent graph in the mixed model analysis is still under investigation.

Finally, a computer program implementing the mixed model Bayesian mapping statistics has been developed. The program is written in FORTRAN language. Although still in its infant form and not necessarily user friendly, with some modifications, the program can analyze real data.

Acknowledgments

This research was supported by the National Institute of Health Grant GM55321 and the USDA National Research Initiative Competitive Grants Programs 97352055075 and 5836359131.

References

Clayton, D. (1999). Estimation in large crossed random-effect models by data augmentation. *J. Royal Stat. Soc. A* 162: 425–436.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *J. Royal Stat. Soc.* 39: 1–38.

Falconer, D. S., and Mackey, T. F. C. (1996). *Introduction to Quantitative Genetics*, 4th ed. Harlow: Longman.

Fernando, R. L., and Grossman, M. (1989). Marker assisted selection using best linear unbiased prediction. *Genet. Sel. Evol.* 21: 467–477.

Geman, S., and Geman, D. (1984). Stochastic relaxation, gibbs distributions and the Bayesian restoration of images. *IEEE Trans. Patt. Anal. Mach. Intell.* 6: 721–741.

Green, P. (1995). Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika* 82: 711–732.

Haldane, J. B. S. (1919). The combination of linkage values, and the calculation of distances between the loci of linked factors. *J. Genetics* 8: 299–309.

Haley, C. S., and Knott, S. A. (1992). A simple regression method for mapping quantitative trait loci in line crosses using flanking markers. *Heredity* 69: 315–324.

Hastings, W. K. (1970). Monte Carlo sampling methods using Markov chains and their applications. *Biometrika* 57: 97–109.

Hoeschele, I., Uimari, P., Grignola, F. E., Zhang, Q., and Gage, K. M. (1997). Advances in statistical methods to map quantitative trait loci in outbred populations. *Genetics* 147: 1445–1457.

Jansen, R. C. (1993). Interval mapping of multiple quantitative trait loci. *Genetics* 135: 205–211.

Kao, C. H., Zeng, Z. B., and Teasdale, R. (1999). Multiple interval mapping for quantitative trait loci. *Genetics* 152: 1203–1216.

Lander, E. S., and Botstein, D. (1989). Mapping mendelian factors underlying quantitative traits using rflp linkage maps. *Genetics* 121: 185–199.

Lynch, M., and Walsh, B. (1998). *Genetics and Analysis of Quantitative Traits.* Sunderland, Mass.: Sinauer Associates.

Martinez, O., and Curnow, R. N. (1992). Estimating the locations and size of the effects of quantitative trait loci using flanking markers. *Theor. Appl. Genet.* 85: 480–488.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953). Equations of state calculations by fast computing machines. *J. Chem. Phy.* 21: 1087–1091.

Satagopan, J. M., Yandell, B. S., Newton, M. A., and Osborn, T. C. (1996). A Bayesian approach to detect quantitative trait loci using Markov chain monte carlo. *Genetics* 144: 805–816.

Sillanpää, M. J., and Arjas, E. (1998). Bayesian mapping of multiple quantitative trait loci from incomplete inbred line cross data. *Genetics* 148: 1373–1388.

Sobel, E., and Lange, K. (1996). Descent graphs in pedigree analysis: Applications to haplotyping, location scores, and marker sharing statistics. *Am. J. Hum. Genet.* 58: 1323–1337.

Van Arendonk, J. A. M., Tier, B., and Kinghorn, B. P. (1994). Use of multiple genetic markers in prediction of breeding values. *Genetics* 137: 319–329.

Wang, C. S., Rutledge, J. J., and Gianola, D. (1993). Marginal inferences about variance components in a mixed linear model using Gibbs sampling. *Genet. Sel. Evol.* 25: 41–62.

Weir, B. S. (1996). *Genetics Data Analysis II.* Sunderland, Mass.: Sinauer Associates.

Xu, S. (1995). A comment on the simple regression method for interval mapping. *Genetics* 141: 1657–1659.

Xu, S. (1998). Iteratively reweighted least squares mapping of quantitative trait loci. *Behav. Genet.* 28: 341–355.

Xu, S., and Yi, N. (2000). Mixed model analysis of quantitative trait loci. *Proc. Natl. Acad. Sci. USA* 97: 14542–14547.

Zeng, Z. B. (1994). Precision mapping of quantitative trait loci. *Genetics* 136: 1457–1468.

9 Finding Genes by Computer: Probabilistic and Discriminative Approaches

Victor V. Solovyev

The more comprehensive and accurate initial computational analysis performed for new genomic sequences, the less time-consuming and costly experimental work will have to be done to determine their functions. For this reason, computational gene identification is an issue of obvious importance as a tool of identifying biologically relevant features (protein coding sequences) that often cannot be found by the traditional sequence database searching technique. This chapter describes statistically based methods for the recognition of eukaryotic genes. We review the structure and significant characteristics of gene components, and discuss recent advances and open problems in gene-finding methodology and its application to sequence annotation of long genomic sequences. Finally, we consider the application of gene expression data for large-scale verification of predicted genes.

9.1 General Features of Eukaryotic Genes

Genes carry and express the hereditary information encoded by segments of nucleic sequence involved in producing protein or RNA molecules. Genetic organization and packaging of eukaryotic genes is fundamentally different from those of prokaryotes. The major differences are large proportion of noncoding DNA (regulatory sequences, introns, repeats, pseudogenes) and the existence of interruptions (introns) that separate different parts of protein coding region in DNA. A typical DNA fragment of protein coding gene includes noncoding regulatory sequences, exons, and introns (figure 9.1).

9.1.1 Gene Expression Steps

The gene is expressed by a several stage process comprising transcription and translation (figure 9.1). Transcription (or pre-mRNA synthesis on DNA template) involves initiation, elongation, and termination steps. RNA polymerase catalyzing RNA synthesis binds a special region (promoter) at the start of the gene and moves along the template, synthesizing RNA, until it reaches a terminator sequence. Post-transcriptional processing of messenger RNA precursors includes capping, 3′-polyadenilation, and splicing. The processing events of mRNA capping and polyA addition take place before pre-mRNA splicing and result in producing the mature mRNA. The mRNA consist of sequences (called exons) that encode the protein product (according to the rules of the genetic code). The gene sequence often includes noncoding regions, called introns, that are removed from the primary tran-

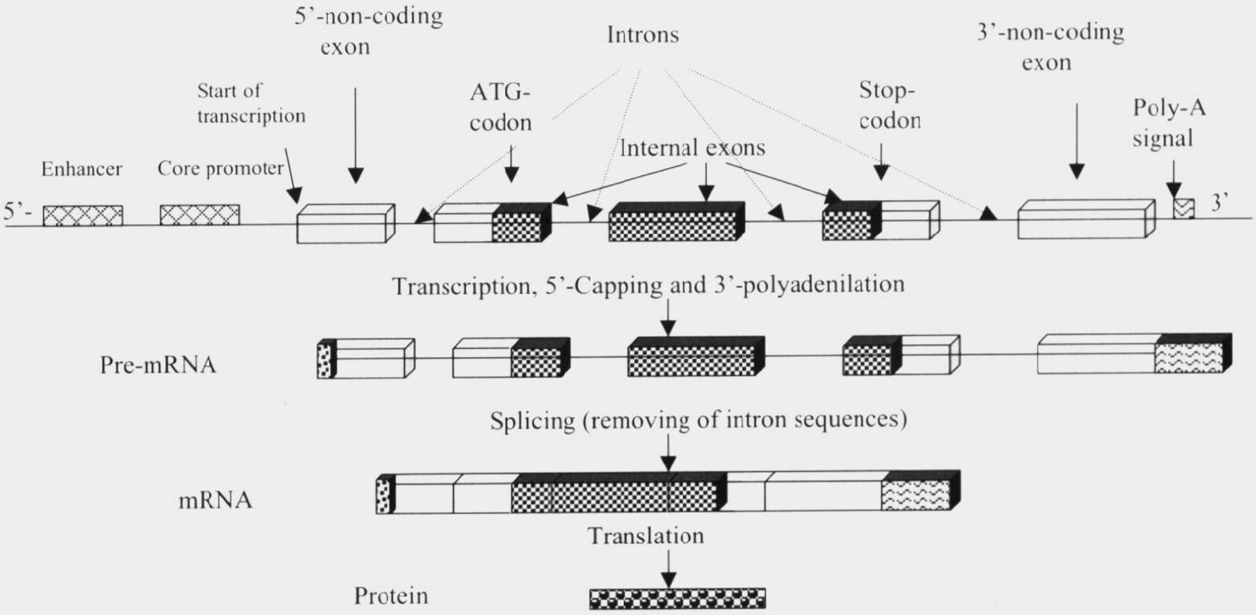


Figure 9.1
Expression stages and structural organization of typical eukaryotic protein-coding gene including associated regulatory regions.

script during RNA splicing. Eukaryotic pre-mRNA is processed in the nucleus and then transported to the cytoplasm for translation. The sequence of mRNA contains a series of triplet codons that interact with the anticodons of aminoacyl-tRNAs (carrying the amino acids) so that the corresponding series of amino acids is incorporated into a polypeptide chain. The small subunit of eukaryotic ribosome binds to the 5′-end of mRNA and then migrates to the special sequence on mRNA (preceding to the start codon) called the ribosome binding site, where it is joined by a large ribosome subunit, forming a complete ribosome. The ribosome initiates protein synthesis at the start codon (AUG in eukaryotes) and moves along the mRNA synthesizing polypeptide chain until it reaches a stop codon sequence (TAA, TGA, or TAG), where release of polypeptide and dissociation the ribosome from the mRNA take place. After that, many proteins undergo post-translational processing (i.e., covalent modifications such as proteolytic cleavage, attachment of carbohydrates and phosphates) before they become functional.

9.1.2 Structural Characteristics

Information about gene structure is accumulated in GenBank and EMBL nucleotide sequence databases. These databases contain annotations of contiguous sequences;

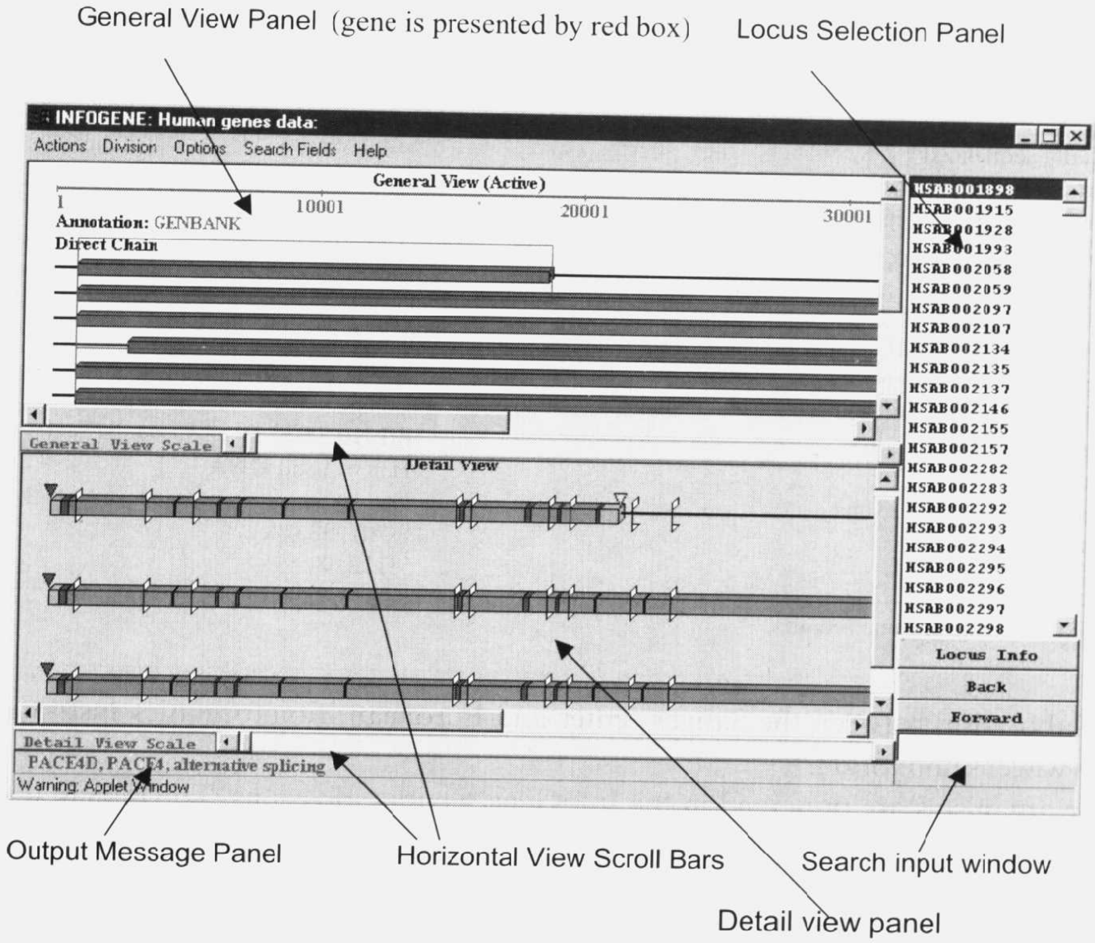


Figure 9.2
InfoGene Java viewer (Seledtsov and Solovyev 1999) presentation of Homo sapiens gene PACE4. This gene has several alternative forms and is described in 17 records of GenBank. Continues sequences regions corresponding different GenBank entries are separated by the vertical bars.

therefore, one gene can be described in dozens of entries with partially sequenced gene regions, alternative splicing forms, or mRNA. The gene-centric database InfoGene (Solovyev and Salamov 1999) contains descriptions of known genes and their basic functional signals extracted from GenBank (Benson et al. 1999). InfoGene also includes all predicted genes for human and *Drosophila* draft genomes and several chromosomes of the *Arabidopsis* genome. InfoGene is realized under JAVA interactive environment system (Seledtsov and Solovyev 1999) that provides visual analysis of known information about complex gene structure (figure 9.2) and search for different gene component and signals. The database is currently available at http://www.softberry.com/infodb.html. A similar project, ENSEMBL, was started

Table 9.1
Structural characteristics of human genes deposited in Genbank (Release 119)

Gene features	Numbers from the Infogen database
CDS/partially sequenced CDS	53435/29404
Exons/partial sequenced exons	83488/21342
Genes/partial sequenced genes	20791/16141
Alternative splicing	2167, 10.4%
Pseudogenes	8.5%
Genes without introns	1552, 7.4%
Number of exons (maximal, average)	117, 5.7
Exon length (range, average)	1–1088, 201.6
Intron length (maximal, average)	259776, 2203.5
Gene length (maximal, average)	401910, 9033
Repeats in genome	41% of total DNA
DNA occupied by coding exons	3%
Donor sites	58707, 98.0%
Acceptor sites	58112, 98.53%

Statistics based on InforGene records.

as a collaboration between the Sanger center and European Bioinformatics Institute (http://www.ensembl.org/).

Major organisms are presented in the InfoGene separate divisions. The human division (based on GenBank 119 release) contains about 21,000 genes, 53,000 coding regions, 83,000 exons, and about 58,000 donor and acceptor splice sites. Table 9.1 shows the major structural characteristics of human genes.

About 41 percent of sequenced human DNA consists of different kinds of repeats. Only about 3 percent of the genome sequence contains protein coding exon sequences. Table 9.2 presents the characteristics of genes in major model organisms such as mouse, *D. melanogaster*, *C. elegans*, *S. cerevisiae*, and *Arabidopsis*.

The gene sizes are often larger in vertebrates, and especially in primates. The average size of an exon is about 190 bp, which is close to the DNA length associated with the nucleosome particle. Human exon sizes are significantly smaller than the gene sizes. There are many exons as short as several bases.

Computational identification of small exons (1–20 bp) cannot be done using the composition based methods that were successful for predicting prokaryote coding regions. Eukaryotic gene prediction approaches should be based on recognition of functional signals encoded in the DNA sequence.

Figure 9.3 illustrates how the same DNA sequences may code several different proteins due to alternative promoters or terminators and alternative splicing. These processes also can significantly complicate computational gene finding.

Table 9.2
Structural characteristics of genes in eukaryotic model organisms

	Mus musculus	*Drosophila melanogaster*	*C. elegans*	*S. cerevisiae*	*Arabidopsis thaliana*
CDS/partial	24527/13060	20314/1510 (20622)	20634/526	12635/1016	31194/1461
Exons/partial	24508/7913	66960/19343 (75935)	122951/38293	13572/13127	145942/42844
Genes/partial	7428/5573	17435/1154 (20622)	19658/1263	12513/1098	28346/1023
Alt. splicing	749, 10%	1785, 10%	1194, 6.1%	598, 4.7%	227, 0.1%
No introns genes	748, 10%	3583, 20%	669, 3%	11070, 88.5%	5776, 19.7%
Number of exons	64, 4.72	50, 3.88	52, 6.1	3, 1.03	78, 5.1
Exon length	1–6642, 207.1	6–10785, 419.5	1–14975, 221.5	1–7471, 1500.0	3–75916, 192.0
Intron length	42573, 818.3	205244, 613.7	19397, 244.0	7317, 300	118637, 174.4
Gene size	150523, 3963	155515, 2854	45315, 2624	14733, 1462	170191, 2027
Donor sites	6225, 96.9%	49592, 98.0%	102872, 99.5%	471, 93.0%	117658, 99.2%
Acceptor sites	5627, 97.5%	49602, 97.9%	102933, 99.7%	475, 95.6%	121917, 96.9%

Description of features is given in table 9.1. For *Drosophila* genes, the numbers in ( ) are taken from *ab initio* computer annotation of *Drosophila* genome by Softberry Inc. (http://www.softberry.com/inf/dros_an.html).

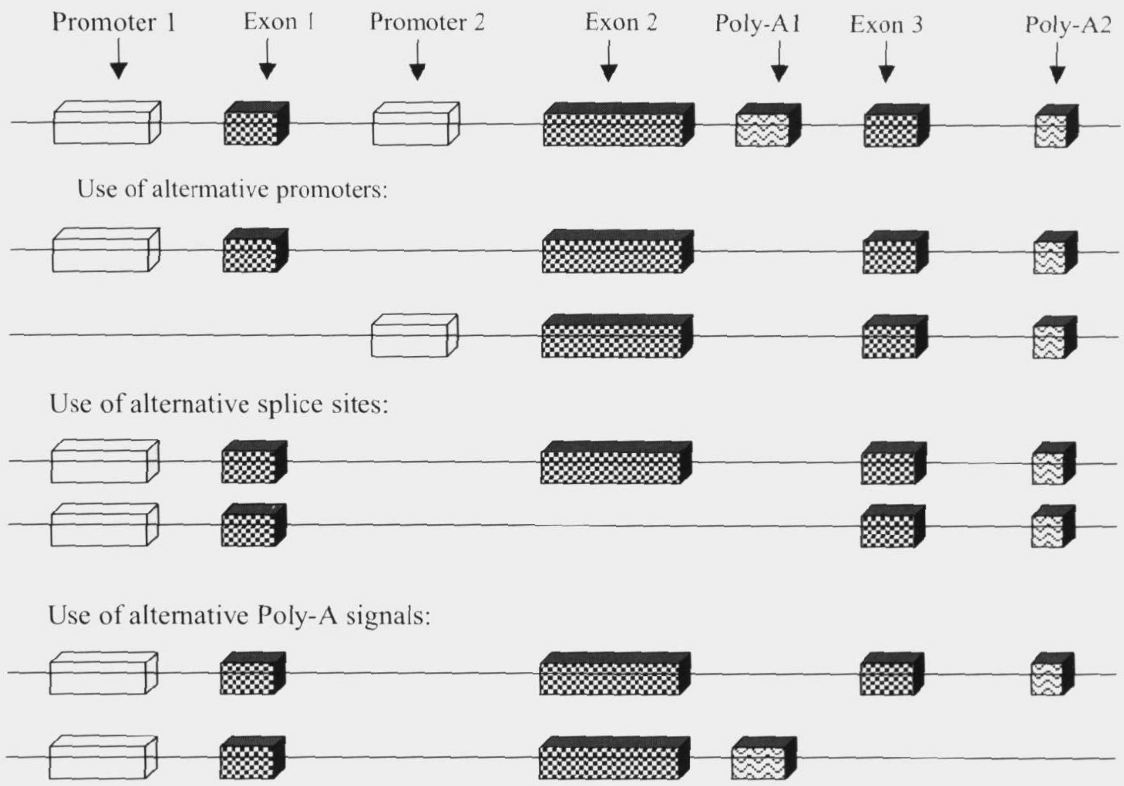


Figure 9.3
DNA region coding alternative gene products.

9.2 Functional Signals Description and Identification

In this section, we focus on several approaches for gene functional signal recognition and some features of these signals used in gene identification. We describe the application of different weight matrices, which usually contain more information about the structure of functional signal than the corresponding consensus sequences, and later elaborate their implementation gene prediction approaches to score potential functional signals.

9.2.1 Position Specific Discrimination

The consensus sequence consists of the most common base at each position of an alignment of binding sites of a particular type. Often it uses a special letters (IUPAC) to indicate the potential presence of more than one nucleotide at a given position. Position weight matrices usually provide better representation of functional signals including quantitative information (Staden 1984; Zhang and Marr 1993; Burge

1997). We can consider the weight matrix as a simple model based on a set of position-specific probability distributions $\{p_s^i\}$, that give the probability of observing a particular type of nucleotide in a particular position of a functional signal (S) sequence. The probability of generating the signal sequence sequence $X(x_1, \ldots, x_k)$ under this model is

$$P(X/S) = \sum_{i=1}^{k} p_{x_i}^i \tag{9.1}$$

where nucleotides of the signal are generated independently. The corresponding model can be constructed for nonsite (N) sequences, $\{\pi_s^i\}$, with the same probability distribution in each position. A discriminative function based on these models is the log likelihood ratio:

$$\text{LLR}(X) = \log P(X/S)/P(X/N) \tag{9.2}$$

It can be written in weight matrix notation, $w(i,s) = \{\log(p_s^i/\pi_s^i)\}$, and

$$\text{Score} = \text{LLR}(X) = \frac{1}{k}\sum_{i=1}^{k} w(i, x_i) \tag{9.3}$$

The other types of weight functions can be used to score the sequence of signal. For example, weights can be generated by some optimization procedures such as perceptron or neural network (Stormo 1982); different position-specific probability distributions $\{p_s^i\}$ might also be considered.

More general types of weight matrix uses position-specific probability distributions $\{p_s^i\}$ of oligonucleotides (instead of nucleotides). Oligonucleotide frequencies are successfully used in Markov chain models, where the probability to generate a particular nucleotide x_i of the signal sequence depends on $k_0 - 1$ previous bases (i.e., depends on oligonucleotide $[k_0 - 1$ base long] ended at the position $i - 1$). Then the probability of generating the signal sequence X is:

$$P(X/S) = p_0 \prod_{i=k_0}^{k} p_{s_{i-1},x_i}^{i-1,i} \tag{9.4}$$

where $p_{s_{i-1},x_i}^{i-1,i}$ is the conditional probability of generating nucleotide x_i in position i given that oligonucleotide s_{i-1} ends at position $i - 1$; p_0 is the probability of generating oligonucleotide $x_1 \ldots x_{k_0-1}$. A simple weight matrix represents an independent mononucleotide model (or 0-order Markov chain), where $k_0 = 1$, $p_0 = 1$ and

$p^{i-1,i}_{x_{i-1},x_i} = p^i_{x_i}$. When we use dinucleotides (first order Markov chain) $k_0 = 2$, $p_0 = p^1_{x_1}$, and $p^{i-1,i}_{x_{i-1},x_i}$ is the conditional probability of generating nucleotide x_i in position i given nucleotide x_{i-1} at position $i-1$. The conditional probability can be estimated from the ratio of observed frequency of oligonucleotide k_0 bases long ($k_0 > 1$) ending at position $i(s_{i-1}, x_i)$ to the frequency of the oligonucleotide $k_0 - 1$ bases long ending at position $i - 1(s_{i-1})$ in a set of aligned sequences of some functional signal:

$$p^{i-1,i}_{s_{i-1},x_i} = \mathrm{f}(s_{i-1}, x_i)/\mathrm{f}(s_{i-1})$$

A model for nonsite sequences for computing $P(X/N)$ is usually based on a 0-order Markov chain with genomic base frequencies (or even equal frequencies [0.25]).

A log likelihood ratio (9.3) with Markov chains was used in a description of promoter, splice sites, and start and stop of translation signals in gene finding programs such as Genscan (Burge and Karlin 1997), Fgenesh (Salamov and Solovyev 1998, 2000) and GeneFinder (Green and Hillier 1998).

A useful discriminative measure taking into account a priori knowledge can be based on computing Bayesian probabilities as components of position-specific distributions $\{p^i_s\}$:

$$\mathrm{P}(\mathrm{S}/o^i_s) = \mathrm{P}(o^i_s/\mathrm{S})\mathrm{P}(\mathrm{S})/(\mathrm{P}(o^i_s/\mathrm{S})\mathrm{P}(\mathrm{S}) + \mathrm{P}(o^i_s/\mathrm{N})\mathrm{P}(\mathrm{N})) \qquad (9.5)$$

where $\mathrm{P}(o^i_s/\mathrm{S})$ and $\mathrm{P}(o^i_s/\mathrm{N})$ can be estimated as position specific frequencies of oligonucleotides o^i_s in the set of aligned sites and nonsites; P(S) and P(N) are the a priori probabilities of site and nonsite sequences, respectively, and o^i_s is a type of the oligonucleotide starting (or ending) in ith position (Solovyev and Lawrence 1993). The probability of a sequence X to belong to a signal, if one assumes independence of oligonucleotides in different positions, is

$$\mathrm{P}(\mathrm{S}/\mathrm{X}) = \sum_{i=1}^{k} P(S/o^i_s)$$

Another empirical discriminator, called "Preference," uses average positional probability to belong to a signal:

$$\mathrm{Pr}(\mathrm{S}/\mathrm{X}) = 1/k \sum_{i=1}^{k} P(S/o^i_s) \qquad (9.6)$$

This measure was used in constructing discriminant functions for the Fgenes gene finding program (Solovyev 1998).

9.2.2 Content Specific Discrimination

To take into account general oligonucleotide composition of a functional region (such as GC-rich promoter sequences) we can use probability distributions and their estimations by oligonucleotide frequencies computed on the whole set of functional signal sequences. Then the Markov chain based probability formula (9.4) of generating the signal sequence X is:

$$P(X/S) = p_0 \sum_{i=k_0}^{k} p_{s_{i-1}, x_i} \quad (9.7)$$

9.2.3 Frame Specific Discrimination

The best discrimination of coding and noncoding sequences in gene prediction approaches was achieved by frame specific recognizers (Claverie and Bougueleret 1986; Claverie et al. 1991; Fickett and Tung 1992). The coding sequence is a sequence of triplets (codons) read continuously from a fixed starting point. Three different reading frames with different codons are possible for any nucleotide sequence (six, if a complementary chain is also considered). It was noted that nucleotides are distributed very unevenly relative to the positions within codons. Therefore, the probability of observing a specific oligonucleotide in coding sequences depends on its position relative to the coding frame (three possible variants), as well as on neighboring nucleotides (Shepherd 1981; Borodovsky et al. 1986; Borodovsky and McIninch 1993). Asymmetry in base composition between codon positions arises due to uneven usage of amino acids and synonymous codons, in addition to the particular structure of genetic code (Guigo 1999). In Markov chain approaches, the frame dependent probabilities $p^{f}_{s_{i-1}, x_i}$ ($f = \{1, 2, 3\}$) are used to model coding regions. The probability of generating a protein coding sequence X is

$$P(X/C) = p_0 \sum_{i=k_0}^{k} p^{f}_{s_{i-1}, x_i} \quad (9.8)$$

where f is equal 1, 2, or 3 for oligonucleotides ending at codon position 1, 2, or 3, respectively.

9.2.4 Prediction Performance Measures

Sensitivity and specificity measures are widely used to characterize the accuracy of an algorithm or a recognition function (Fickett and Tung 1993; Snyder and Stormo 1993;

1994; Dong and Searls 1994). Let us have S sites (positive examples) and N nonsites (negative examples). By applying the recognition function, we identify correctly T_p sites (true positives) and T_n nonsites (true negatives). At the same time, F_p (false positives) sites are wrongly classified as nonsites and F_n (false negative) nonsites are wrongly classified as sites. Note that $S = T_p + F_n$ and $N = T_n + F_n$. Sensitivity (S_n) measures the fraction of the true positive examples that are correctly predicted: $S_n = T_p/(T_p + F_n)$. Specificity (S_p) measures the fraction of the predicted examples that are correct: $S_p = T_p/(T_p + F_p)$. When we see only one value of accuracy estimation, it means the average accuracy of sites and nonsites is a true prediction: $AC = 0.5(T_p/S + T_n/N)$. The more general single measure (correlation coefficient) takes into account a possible difference in the sizes of site and nonsite sets (Matthews 1975):

$$CC = (T_pT_n - F_pF_n)/\sqrt{(T_p + F_p)(T_n + F_n)(T_p + F_n)(T_n + F_p)}$$

9.2.5 Fisher's Linear Discriminant

The linear discriminant analysis approach provides a method to select a "best" set of an objects features and combine them in a discriminant function that yields an output that is an estimate of the class membership of this object. We assume that each given sequence fragment can be described by a vector $\dot{x}$ of p characteristics $(x_1, x_2, \ldots, x_p)$, which can be measured. The procedure of linear discriminant analysis is to find a linear combination of the measures (called the linear discriminant function or LDF) that provides maximum discrimination between sites sequences (class 1) and nonsite examples (class 2). The LDF

$$Z = \sum_{i=1}^{p} a_i x_i$$

classifies (X) into class 1 if $Z > c$ and into class 2 if $Z < c$. The vector of coefficients $(\alpha_1, \alpha_2, \ldots \alpha_p)$ and the threshold constant c are derived from the training set by maximizing the ratio of the between-class variation of z to within-class variation (or minimizing expected probability of misclassification) and are equal to (Duda and Hart 1973; Afifi and Aizen 1979)

$$\dot{a} = s^{-1}(\dot{m}_1 - \dot{m}_2)$$

$$c = \dot{a}(\dot{m}_1 + \dot{m}_2)/2 + \ln \frac{p_2^0}{p_1^0}$$

where $\dot{m}_i$ are the sample mean vectors of characteristics for class 1 and class 2, respectively; s is the pooled covariance matrix of characteristics,

$$s = \frac{1}{n_1 + n_2 - 2}(s_1 + s_2)$$

s_i is the covariation matrix, p_i^0 is the prior probability, and n_i is the sample size of class i. Based on these equations, we can calculate the coefficients of LDF and the threshold constant c using the characteristics of site and nonsite sequences from the training sets, and we can then test the accuracy of LDF on the test set data. This classification actually assigns a feature vector $\dot{x}$ to the category of the nearest mean measure, the squared Mahalanobis distance

$$\dot{D}^2 = (\dot{x} - \dot{m}_i)s^{-1}(\dot{x} - \dot{m}_i)$$

from $\dot{x}$ to each of the mean vectors m_i. The significance of a given characteristic or a set of characteristics can be estimated by the Mahalonobis distance between two classes:

$$\dot{D}^2 = (\dot{m}_1 - \dot{m}_2)s^{-1}(\dot{m}_1 - \dot{m}_2)$$

which is computed based on values of the characteristics in the training sequences of classes 1 and 2. To find discriminating sequence features, many possible characteristics, such as score of weigh matrices, distances, oligonucleotide preference at different subregions, and so on, are generated. Selection of the subset of significant characteristics q (among the tested p) is performed by a step-wise discriminant procedure including only characteristics, which significantly increases the Mahalonobis distance. The procedure to test this significance uses the fact that the quantity:

$$F = \frac{n_1 + n_2 - p - 1}{p - q} \frac{n_1 n_2 (D_p^2 - D_q^2)}{(n_1 + n_2)(n_1 + n_2 - 2) + n_1 n_2 D_q^2}$$

has an $F\ (p - q, n_1 + n_2 - p - 1)$ distribution when testing hypothesis H_0: $\Delta_p^2 = \Delta_q^2$, where Δ_m^2 is the population Mahalonobis distance based on m variables (Afifi and Aizen 1979).

9.2.6 Quadratic Discriminant Analysis

Classical linear discriminant analysis often assumes the probability model in which the observations for classes have different means, but a common covariation matrix. The feature space is partitioned by hyperplane optimally separating observations of different classes. To classify groups having different covariation matrices, one can use

the quadratic discriminant analysis (QDA). Quadratic discriminant analysis provides a curved boundary in multidimensional feature space. Maximum discrimination between the two classes is achieved with the quadratic discriminant function QDF:

$$QDF = \log \frac{p_1^0}{p_2^0} - \frac{1}{2}(D_1^2 - D_2^2) - \frac{1}{2}\log\frac{|S_1|}{|S_2|}$$

where D_i^2 is Mahalonobis distance from an object to the mean and Si is the covariation matrix of class i ($i = 1, 2$). Quadratic discriminant function might provide a more effective discrimination, but will require a larger learning set of observations to accurately define its larger set of parameters. Such an approach was used in exon prediction method developed by Zhang (1997) to improve the accuracy of the linear discriminant exon predictor (Solovyev et al. 1994).

9.2.7 Splice Sites Conservative Features

The precise removal of introns from mRNA precursors is mainly defined by the highly conserved sequences near the ends of introns (Breatnach and Chambon 1981; Wieringa et al. 1983). The donor (or 5′-splice site) is characterized by a sequence of eight nucleotides AG|GTRAGT. The acceptor (or 3′-splice site) possesses a sequence of four nucleotides preceded by a pyrimidine rich region: YYTTYYYYYYNC|AGG (Senapathy et al. 1990). The third less conserved intron sequence (branch site), of about 5–8 nucleotides and containing an adenosine residue, usually lies between 10 and 50 nucleotides upstream of the acceptor splice site.

The vast majority of introns contains invariant GT and AG dinucleotides at their termini excised from pre-mRNA by the spliceosome, including U1, U2, U4/U6, and U5 snRNPs (Breatnach et al. 1978; Breatnach and Chambon 1981; Nilsen 1994). A rare type of splice pair, the AT-AC, has also been discovered. It is processed by related but different splicing machinery (Jackson 1991; Hall and Padget 1994). For the AT-AC group, different conserved positions have been noticed: |ATATCCTTT for the donor site and YAC| for the acceptor site (Deitrich et al. 1997; Sharp and Burge 1997; Wu and Krainer 1997).

Burset et al. (2000) have recently done a comprehensive investigation of canonical and noncanonical splice sites. They applied ESTs and high-throughput genomic (HTG) sequences to analyze 43,437 pairs of exon-intron boundaries and their sequences from InfoGene (Seledtsov and Solovyev 1999) database, including all annotated genes in mammalian genomic sequences. Of the 43,437 pairs of donor and acceptor splice sites (splice pairs), 1,215 were annotated as nonstandard donor sites (2.80 percent), and 1,027 were annotated as nonstandard acceptor sites (2.36 percent). Forty-one thousand seven hundred and sixty-seven splice pairs (96.18 percent)

Table 9.3
Splice sites sequences presented in the SpliceDB (Burset, Seledtsov, and Solovyev 2001)

Sequences of splice pairs	Canonical	Noncanonical
Mammals		
Original from GenBank	41722 (**96.27**%)	1615 (**3.73**%)
EST supported	22374 (**98.07**%)	441 (**1.93**%)
EST supported + corrected	22199 (**98.71**%)	290 (**1.29**%)
Human		
Original from GenBank	27486 (**96.55**%)	982 (**3.45**%)
EST supported	15384 (**98.33**%)	261 (**1.67**%)
EST supported + corrected	15263 (**98.89**%)	171 (**1.11**%)
HTG supported		156

contained the standard splice site pair GT-AG. Analysis showed that of 1,615 noncanonical pairs, 441 were supported by EST (27.3 percent) and just 290 (18 percent) were supported by EST after removing potential annotation errors and examples with ambiguity in the position of the splice junction (table 9.3).

Analysis of human noncanonical splice pairs that have corresponding EST and HTG sequences shows that all human EST-supported GC-AG cases having HTS matches were supported (39 cases). Thirty-one errors were found damaging the standard splice pairs (seven cases had one or both intronic GenBank sequences completely unsupported by HTS, whereas eight cases had intronic GenBank sequences supported; there was a gap between exonic and intronic parts, and in the end, 16 cases had small errors, such as insertions, deletions, or substitutions). Three of five observed AT-AC pairs were correctly annotated in the original noncanonical set; two were recovered from errors. Two more cases were annotated as introns, but in HTS, the exonic parts were continuous (accession numbers: U70997 and M13300).

This analysis shows that the overwhelming majority of splice sites contain conserved dinucleotides GT-AG (99.2 percent). The other major group includes GC-AG pairs (0.62 percent), the alternative splicing machine group AC-AT (about 0.08 percent), and a very small number of other noncanonical splice sites (about 0.03 percent). Therefore, gene-finding approaches using just standard GT-AG splice sites can potentially correctly predict 97 percent genes (if we assume four exons per gene, on average). Including the GC-AG splice pair will increase this level to 99 percent. Twenty-two thousand two hundred and fifty-three verified examples of canonical splice pairs were presented in a SpliceDB database, which is available at http://genomic.sanger.ac.uk (Burset et al. 2000). It also includes 1,615 annotated and 292 EST-supported and shift-verified noncanonical pairs. The weight matrices and consensus sequences for the major group of splice sites are presented in figure 9.4.

a) GT-AG group (canonical splice sites): 22199 examples

$$M_{70}A_{60}G_{80}|GTR_{95}A_{71}G_{81}T_{46}$$

	-3	-2	-1	1	2	3	4	5	6
A	34.0	60.4	9.2	0.0	0.0	52.6	71.3	7.1	16.0
C	36.3	12.9	3.3	0.0	0.0	2.8	7.6	5.5	16.5
G	18.3	12.5	80.3	100	0.0	41.9	11.8	81.4	20.9
U	11.4	14.2	7.3	0.0	100	2.5	9.3	5.9	46.2

$$Y_{73}Y_{75}Y_{78}Y_{79}Y_{80}Y_{79}Y_{78}Y_{81}Y_{86}Y_{86}NC_{71}AG|G_{52}$$

	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1	1
A	9.0	8.4	7.5	6.8	7.6	8.0	9.7	9.2	7.6	7.8	23.7	4.2	100	0.0	23.9
C	31.0	31.0	30.7	29.3	32.6	33.0	37.3	38.5	41.0	35.2	30.9	70.8	0.0	0.0	13.8
G	12.5	11.5	10.6	10.4	11.0	11.3	11.3	8.5	6.6	6.4	21.2	0.3	0.0	100	52.0
U	42.3	44.0	47.0	49.4	47.1	46.3	40.8	42.9	44.5	50.4	24.0	24.6	0.0	0.0	10.4

b) GC-AG group: 126 examples

$$M_{83}A_{89}G_{98}|GCA_{87}A_{84}G_{97}T_{71}$$

	-3	-2	-1	1	2	3	4	5	6
A	40.5	88.9	1.6	0.0	0.0	87.3	84.1	1.6	7.9
C	42.1	0.8	0.8	0.0	100	0.0	3.2	0.8	11.9
G	15.9	1.6	97.6	100	0.0	12.7	6.3	96.8	9.5
U	1.6	8.7	0.0	0.0	0.0	0.0	6.3	0.8	70.6

$$Y_{75}Y_{72}Y_{78}Y_{79}Y_{77}Y_{80}Y_{66}Y_{78}Y_{85}Y_{84}NC_{68}AG|G_{63}$$

	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1	1
A	11.1	12.7	3.2	4.8	12.7	8.7	16.7	16.7	12.7	9.5	26.2	6.3	100	0.0	21.4
C	36.5	30.9	19.1	23.0	34.9	39.7	34.9	40.5	40.5	36.5	33.3	68.2	0.0	0.0	7.9
G	9.5	10.3	15.1	12.7	8.7	9.5	16.7	4.8	2.4	6.3	13.5	0.0	0.0	100	62.7
U	38.9	41.3	58.7	55.6	42.1	40.5	30.9	37.3	44.4	47.6	27.0	25.4	0.0	0.0	7.9

c) AT-AC group: 8 annotated examples + 2 examples recovered from annotation errors

$$S_{90}|ATA_{100}T_{100}C_{100}C_{100}T_{100}T_{90}T_{70}$$

$$T_{70}G_{50}C_{70}NC_{60}AC|A_{60}T_{60}$$

AC002397 **TGCCAAGATG**|atatccttgtgt ctgtctgctcac|**CTTGGAGAAG**
AC004976 **GAAAGAACCC**|atatcctttctg actacttcatac|**AAAACAGTCA**
AF136179 **TATGGTAGAG**|atatcctttact actgtttcggac|**ATTGACCAAA**
AL021578 **ACGCTGAACC**|atatcctttggg ttaaccgctcac|**TGGCCCAGCT**
L10295 **ATTGGTGAAG**|atatccttttag aatcattactac|**ATGTGAATCC**
U39892 **AGATTAGAGA**|atatcctttctt aactgccagcac|**ATTTTGTCAG**
U47924 **TGCCAAGATG**|atatccttctgc aaccctcctcac|**CTTGGAGAAG**
U53004 **GGAAGTGGTC**|atatccttcctg aactctgcacac|**GAAGCTCACG**

Figure 9.4
Consensus sequences and weight matrices for major groups of splice site pairs. Numbering splice site positions is provided relative to the splice junction along the gene sequence.

Table 9.4
Significance of various characteristics for discimination of donor splice sites

Characteristics	1	2	3	4	5	6	7
Individual D^2	9.3	2.6	2.5	0.1	1.5	0.01	0.4
Combined D^2	9.3	11.8	13.6	14.9	15.5	16.6	16.8

1, 2, 3 are the triplet preference (13) of consensus (_4 __ +6), intron G rich (+7 __ +50) and coding regions (_30 __ _5), respectively. 4 is the number of significant triplets in the consensus region. 5 and 6 are the octanucleotide preference for being coding 54 bp region on the left and for being intron 54 bp region on the right of donor splice site junction. 7 is the number of G bases, GG doublets, and GGG triplets in +6 __ +50 intron G rich region.

9.2.8 Computational Recognition of Splice Sites

Computational analysis of splice site sequences demonstrates that their consensuses are slightly specific for distant classes of organisms (Senapathy et al. 1990; Mount 1993) and that some important information is encoded by the sequences outside the short conserved regions. There were several attempts to develop accurate splice site identification algorithms based on consensus sequences or weight matrices, which take into account the information about open reading frames, free energy of base pairing of RNA with snRNA, and other sequence features. These approaches reached the accuracy of about 80 percent for the prediction splice site positions (Nakata et al. 1985; Gelfand 1989). More accurate prediction was achieved by neural network algorithms (Lapedes et al. 1988; Brunak et al. 1991). To demonstrate what sequence features can help identify authentic splice sites, we will describe a simple method using a linear discriminant function (Solovyev and Lawrence 1993; Solovyev et al. 1994).

Donor Splice Site Recognition To test the significance of different sequence features by liner discriminant approach described in section 9.2.5, seven characteristics were selected for donor splice site identification. In table 9.4, we can see the Mahalonobis distances showing the significance of each characteristic. The strongest characteristic of donor sites is triplet composition in consensus region ($D^2 = 9.3$), in the adjacent intron region ($D^2 = 2.6$), and in coding region ($D^2 = 2.5$). Other significant characteristics are a number of significant triplets in conserved consensus region; the number of G bases, GG doublets, and GGG triplets; and the octaplet composition of the coding and intron regions.

A rigorous testing of several splice site prediction programs on the same sets of new data demonstrated that the linear discriminant function (implemented in SPL program: http://www.genomic.sanger.ac.uk) provides the most accurate local donor site recognizer (table 9.5) (Milanesi and Rogozin 1998).

Table 9.5
Comparing the prediction accuracy of local donor splice site recognizers

Method/program	False positives	False negatives	CC	Reference
Weight matrix	5.0%	20%	0.22	Guigo et al. 1991
Neural network (NETGENE)	16.3%	6.7%	0.35	Brunak et al. 1991
Discriminant analysis (SPL)	22.0%	2.3%	0.51	Solovyev et al. 1994

The accuracy is averaged for three tested sets.

A single weight matrix provides less accurate recognition than more sophisticated approaches, but it can be easily recomputed for new organisms and is very convenient to use in probabilistic HMM gene prediction methods. Using maximal dependence decomposition procedure (Burge 1998), we constructed five donor recognition weight matrices for different subsets of splice site sequences. The subclassification of donor signals and the matrices constructed, based on 22,306 EST supported splice sites, are presented in figure 9.5. Performance of these matrices comparing with the other methods was estimated on the Burset and Guigo (1996) data set (figure 9.6). It shows that several weight matrices provide better splice site discrimination than just one. However, their discriminative power is similar to the triplet matrix and lower for most levels of sensitivity than the liner discriminant function of the SPL program.

Acceptor Splice Site Recognition The performance of acceptor site recognition (Rogozin and Milanesi 1997) by different computational methods is presented in table 9.6. We can see that acceptor site recognition accuracy is lower than the accuracy of predicting donor sites. The linear discriminant function (Solovyev et al. 1994) implemented in the SPL program demonstrates the higher accuracy.

Burge (1998) demonstrated that the first order Markov chain model formula (9.11) based on dinucleotide frequencies of [−20, +3] acceptor site region gives slightly better discrimination than the simple weight matrix model. Such a model was implemented in the Genscan gene prediction method (Burge and Karlin 1997). Recently, Thanaraj (2000) evaluated several splice site recognitions. Among them, the SPL program remains the best local recognizer. Of course, complex gene prediction systems (HMM gene, Genscan, Fgenes, Fgenesh, and some intermediate approaches such as NetGene2) using a lot of global information about optimal exon (or splice site) combination will have a better accuracy level. However, they cannot be applied to study possible alternative splice sites in a particular gene. Local recognizers might be useful for such tasks.

	-4	-3	-2	-1	1	2	3	4	5	6	7	8	9	10
A	27.6	34	60.5	9.2	0	0	52.5	71.3	7.1	16.1	27.4	20.8	20.1	19.9
T	28.7	36.2	12.8	3.3	0	0	2.8	7.6	5.5	16.6	21.1	27.1	28.9	25.5
G	24	18.3	12.4	80.3	100	0	42.2	11.8	81.5	21	32.7	25.5	26.7	29.3
C	19.7	11.4	14.3	7.2	0	100	2.5	9.3	5.9	46.3	18.7	26.5	24.2	25.3

G_5H_{-1}
4029 sites

	A	T	G	C
-3	32.2	29.1	21.6	17.1
-2	42.9	30.9	17.2	9.1
-1	46.6	16.6	0	36.8
3	54.5	0.4	44.7	0.4
4	90.4	2.9	3	3.6
6	5.6	7.9	6.3	80.2

H5
3930 sites

	A	T	G	C
-3	39.4	40.9	14.2	5.5
-2	83.6	4.8	4.9	6.7
-1	1.9	0.7	95.9	1.5
3	79.7	1.3	16.8	2.2
4	53.2	24.6	9.6	12.7
5	38.1	30	0	31.9
6	21	18.3	30.8	29.8

$G_5G_{-1}B_{-2}$
5474 sites

	A	T	G	C
-3	31.4	28.9	16.9	22.8
-2	0	23.6	32.2	44.3
3	45.3	2	51.8	0.9
4	81.5	3.1	8	7.4
6	14.2	16.5	18.1	51.2

$G_5G_{-1}A_{-2}V_6$
5183 sites

	A	T	G	C
-3	34.4	43.4	19.3	2.9
3	47	4.4	46	2.5
4	67	4.7	18.5	9.8
6	30.9	30.5	38.5	0

$G_5G_{-1}A_{-2}T_6$
2636 sites

	A	T	G	C
-3	34.3	39.8	21	5
3	36.2	7.2	47	9.7
4	56	4.6	23.1	16.2

Figure 9.5
Classification of donor splice sites by several weight matrices reflecting different splice site groups. Classification computed on 21,252 verified splice sites from SpliceDB. Each cell represents the frequency (in percent) of a particular base in some position of donor site. H is (A, C, or T), B is (C, G, or T), V is (A, C, or G).

9.2.9 PolII Promoter Recognition

Because each eukaryotic polymerase II promoter has a unique selection and arrangement of regulatory elements providing a unique program of gene expression, the computational identification of promoter sequences in genomic DNA is an extremely difficult problem (see chapter 2). Here we consider a version of promoter recognition program TSSW (Solovyev and Salamov 1997), several modules of which are implemented in the gene prediction program FGENES (Solovyev 1997). In the last version of TSSW, it was assumed that TATA+ and TATA− promoters have very different sequence features, so these groups were analyzed separately. Potential

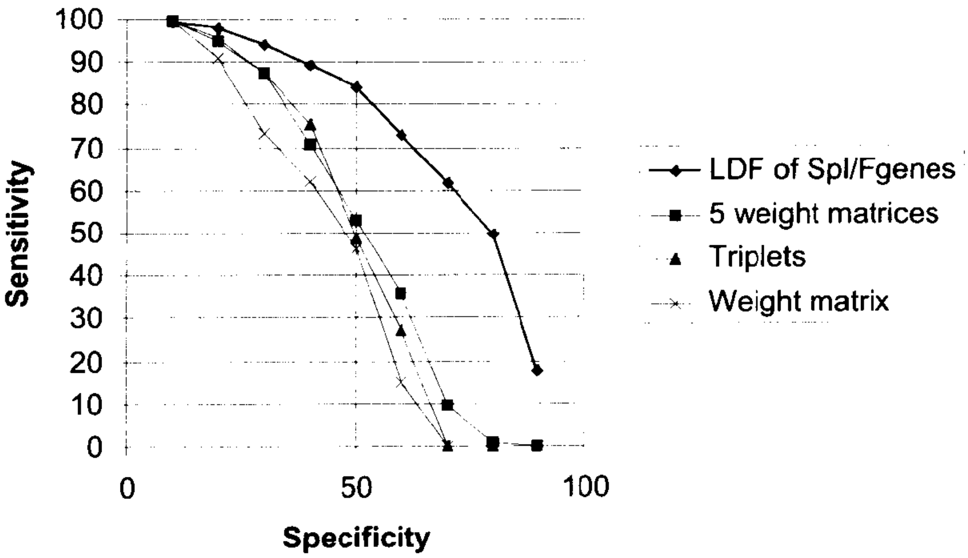


Figure 9.6
Comparing accuracy of donor splice site recognizers: single weight matrix, five weight matrices, matrix of triplets, linear discriminant function.

Table 9.6
Comparing the prediction accuracy of local acceptor splice site recognizers

Method/program	False positives	False negatives	CC	Reference
Weight matrix	2.3%	53%	0.13	Guigo et al. 1992
Consensus MAG/GURAGU	6.0%	18%	0.27	Mount 1982
Consensus MAG/GURAGU	6.0%	18%	0.27	Mount 1982
Five consensuses	4.2%	15%	0.31	Rogozin and Milanesi 1997
Neural network (NETGENE)	25.0%	2.7%	0.51	Brunak et al. 1991
Discriminant analysis (SPL)	10.0%	3.0%	0.56	Solovyev et al. 1994

The accuracy is averaged for three tested sets.

TATA+ promoter sequences were selected by the value of the score of Bucher TATA box weight matrix (Bucher 1990) with the threshold close to the minimal score value for the TATA+ promoters in the learning set. Such a threshold divides the learning sets of known promoters into approximately equal groups. Selected significant characteristics of both groups found by discriminant analysis are presented in table 9.7. This analysis demonstrated that TATA− promoters have much weaker general features than TATA+ promoters. Probably TATA− promoters possess more gene specific structure; they will be extremely difficult to predict by any general-purpose methods.

Table 9.7
Characteristics of promoter sequences used by TSSW programs for identification of TATA+ and TATA− promoters

Characteristics	D^2 for TATA+ promoters	D^2 for TATA− promoters
Hexaplets −200 − −45	2.6	1.4 (−100 − −1)
TATA box score	3.4	0.9
Triplets around TSS	4.1	0.7
Hexaplets +1 − +40		0.9
Sp1-motif content		0.9
TATA fixed location	0.7	
CpG content	1.4	0.7
Similarity −200 − −100	0.3	0.7
Motif Density (MD) −200 − +1	4.5	3.2
Direct/Inverted MD −100 − +1	4.0	3.3 (−100 − −1)
Total Mahalonobis distance	11.2	4.3
Number promoters/nonpromoters	203/4000	193/74000

For each position of a given sequence, the TSSW program evaluates the occurrence of TSS using two linear discriminant functions (for TATA+ and TATA− promoters) with characteristics computed in the (−200, +50) region around the given position. If we find a TATA-box (using TATA-box weight matrix) in this region, then we compute the value of LDF for TATA+ promoters, otherwise the value of LDF for TATA-less. Only one prediction with the highest score of LDF and greater than some threshold will be selected within any 300-bp region. If we observe a lower scoring promoter predicted by the TATA− LDF near a higher scoring promoter predicted by TATA+ LDF, then the first prediction is also displayed as a potential enhancer region.

Figure 9.7 shows an example of TSSW program results for the sequence of the human connexin 32 (GJB1) gene (GenBank accession number L47127). The TSSW predicts one enhancer at position 246 and one potential TSS at position 428, with corresponding TATA-box at the position 388. GenBank annotation based on experimental data shows real TATA-signal in positions 389–394. TSSW also optionally lists all potential TF binding sites around predicted promoters or enhancers (figure 9.7). It outputs the position, the strand (±), TRANSFAC identifier, and the consensus sequences of found sites. The information about these sites may be of interest for researchers studying the transcription of a particular gene.

Due to a high false positive rate of promoter prediction in long genomic sequences, they are more useful when we can remove some predictions about the positions of coding regions. The TSSW was additionally tested on the several GenBank entries that have information about experimentally verified TSS and were not included in the

```
tssw  Wed Dec 20 20:02:08 PST 2000
> Homo sapiens connexin 32 (GJB1) gene
 Length of sequence-       950
 Thresholds for TATA+ promoters -  0.45, for TATA-/enhancers -  3.70
     2 promoter/enhancer(s) are predicted
 Enhancer Pos:     246 LDF- 14.25
 Promoter Pos:     428 LDF-  0.64 TATA box at     388     18.68
 Transcription factor binding sites:
for position -      246
   179 (+) CHICK$ACRA   CCGCCC
   241 (-) CHICK$ACRA   CCGCCC
   220 (-) CHICK$ACRA   CCGCCC
   188 (+) MAIZE$ADH1   CGTGG
    26 (+) Y$ADH2_01    TCTCC
   168 (+) Y$ADH2_01    TCTCC
    42 (-) Y$ADH2_01    TCTCC
   215 (+) HS$APOE_08   GGGCGG
   236 (+) HS$APOE_08   GGGCGG
   184 (-) HS$APOE_08   GGGCGG
   243 (-) HS$APOE_09   GCCCGCCC
   146 (+) MOUSE$MCK_   cccaaCACCTGCtgcctgagcc
    87 (-) RAT$EAI_09   GTCAG
    14 (+) RAT$AFEP_0   TGTCCT
    50 (+) Y$GAL1_10    AGCCT
   116 (+) Y$GAL1_10    AGCCT
   226 (-) HS$BG_01     ccaCACCCg
   214 (+) CHICK$BAG_   GGGGCGGG
   185 (-) CHICK$BAG_   GGGGCGGG
   ..................................
   for promoter at position -      428
   179 (+) CHICK$ACRA   CCGCCC
   423 (-) CHICK$ACRA   CCGCCC
   312 (-) CHICK$ACRA   CCGCCC
   241 (-) CHICK$ACRA  • CCGCCC
   220 (-) CHICK$ACRA   CCGCCC
   389 (+) CHICK$BAC_   TATAA
   188 (+) MAIZE$ADH1   CGTGG
   168 (+) Y$ADH2_01    TCTCC
   215 (+) HS$APOE_08   GGGCGG
   236 (+) HS$APOE_08   GGGCGG
   307 (+) HS$APOE_08   GGGCGG
   418 (+) HS$APOE_08   GGGCGG
   184 (-) HS$APOE_08   GGGCGG
   243 (-) HS$APOE_09   GCCCGCCC
   357 (+) BPV1$BPV1_   ACCagtaatGGT
   368 (-) BPV1$BPV1_   ACCagtaatGGT
   357 (+) BPV1$BPV1_   ACCgttgccGGT
   368 (-) BPV1$BPV1_   ACCgttgccGGT
   357 (+) BPV1$BPV1_   ACCgtcttcGGT
   368 (-) BPV1$BPV1_   ACCgtcttcGGT
   417 (+) MOUSE$A21C   gcccagccctcccATTGGtggagacg
   146 (+) MOUSE$MCK_   cccaaCACCTGCtgcctgagcc
   381 (+) AD$E3_06     gggcagggTATAActcacctga
   ..................................
```

Figure 9.7
Results of promoter prediction by TSSW program in human connexin 32 (GJB1) gene (GenBank accession number L47127).

Table 9.8
The TSSW predictions on some GenBank entries with experimentally verified TSS

Gene	GenBank accession number	Length (bp)	True TSS	Predicted TSS	Number of false positives
CXCR4	AJ224869	8747	2632	2631	4
HOX3D	X61755	4968	2280	2278	2
DAF	M64356	2003	733	744	1
GJB1	L47127	950	404	428	0
ID4	AF030295	1473	1066	1081	1
C inhibitor	M68516	15571	2200	2004	4
MBD1	AJ132338	2951	1964	1876	1
Id -3	X73428	2481	665	663	0

learning set (table 9.8). The lengths of sequences varied from 950 to 28,438 bp, with a median length of 2,938 bp. All true TSS in these sequences can be considered as correctly predicted, with an average of 1.5 false positives per sequence or 1 false positive per 3,340 bp. The distances between true TSS and the correctly predicted ones varied from exact matching to 196 bp, with the median deviation of about 15 bp, which means that half of the predictions are close to the experimental mapping of TSS with the estimated precision of ± 5 bp (Perier et al. 2000).

The above prediction algorithm uses the propensities of each TF binding site independently, not taking into account their mutual orientation and positioning. At the same time, the transcription regulation is a highly cooperative process, involving the simultaneous binding of several transcription factors to their corresponding sites. In future algorithms we should analyze patterns of regulatory sequences, where mutual orientation and location of individual regulatory elements are necessary.

9.2.10 Recognition of PolyA Signals

A 3′-untranslated region (3′UTR) has a diversity of cytoplasmic functions affecting the localization, stability, and translation of mRNAs (Decker and Parker 1995). Practically all eukaryotic mRNAs undergo 3′-end processing, which involves endonucleotide cleavage followed by polyadenylation of the upstream cleavage product (Wahle 1995; Manley 1995). The formation of large RNA-protein complexes is essential for 3′-end processing (Wilusz et al. 1990). RNA sequences directing the binding of specific proteins are usually poorly conserved and often recognized in a cooperative fashion (Wahle 1995). Therefore, the approaches for poly-A signal identification use statistical characteristics of poly-A regions, which can reflect some unknown functional sequences.

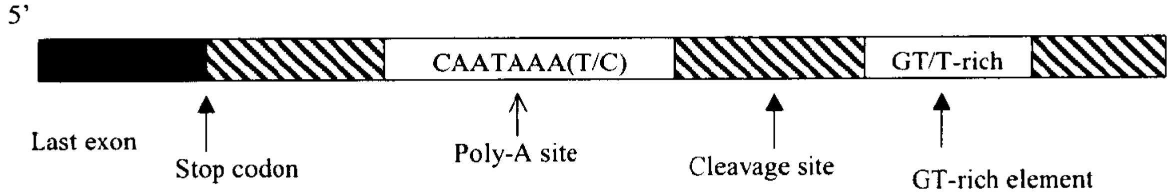


Figure 9.8
Basic structure of poly-A signal sequences.

There are three types of basic RNA sequences defining a 3′-processing site (Wahle 1995; Proudfoot 1991) (figure 9.8). The most conserved is the hexamer signal AAUAAA (polyA signal), situated 10–30 nucleotides upstream of the 3′-cleavage site. About 90 percent of sequenced mRNAs have a perfect copy of this signal. Two other types, the upstream and the downstream elements, are poorly conserved and characterized. Downstream elements are frequently located within approximately 50 nucleotides 3′ of the cleavage site and often GU or U rich (Wahle and Keller 1992). Comparing their sequences, McLachlan et al. (1985) suggest a possible consensuse of one downstream element: YGUGUUYY. The efficiency of polyadenylation in a number of genes can be also increased by generally U-rich sequences upstream of AAUAAA (Wahle 1995).

A few computer programs were developed to identify 3′-processing. Yada et al. (1994) analyzed human DNA sequences in the vicinity of the poly-A signal, trying to distinguish them from other AATAAA sequences nonactive in polyadenylation (pseudo polyA signals). They found that C frequently appears on the upstream side of the AATAAA signal and T or C often appears on the downstream side, generating an extended consensus of poly-A signal: CAATAAA(T/C). Kondrakhin et al. (1994) constructed a generalized consensus matrix using 63 sequences of cleavage/polyadenylation sites in vertebrate pre-mRNA. The matrix elements were the absolute frequencies of triplets at each site position. Using this matrix for recognition of polyadenylation regions produces a very high number of false positive predictions.

A LDF recognition function for poly-A signal was developed by Salamov and Solovyev (1997). The prediction algorithm was realized in the POLYAH program. It searches for the AATAAA pattern by using weight matrix. After it finds the pattern, it computes the value of the linear discriminant function, defined by seven sequence characteristics around this position. The POLYAH program has been tested on the sequence of Ad2 genome, where for eight correctly identified sites, it predicts only four false sites.

Further improvement of poly-A recognition was reached in using a pair of quadratic discriminant function in the Polyadq program (Tabaska and Zhang 1999).

This program outperformed the POLYAH detection method and is the first that can detect significant numbers of ATTAAA-type signals.

9.3 ORF, Exon, and Single Gene Prediction Approaches

The first generation of computational gene finding programs searched for open reading frames with organism-specific codon usage (Staden and McLachlan 1982). These approaches worked successfully for bacterial genes (Staden 1984; Borodovsky et al. 1986), but short eukaryotic exons and spliced eukaryotic genes require algorithms taking into account additional information about functional signals. One application of such approaches is useful for predicting coding regions or coding ORF in partially or completely sequenced mRNA(EST) sequences. Several HMM-based predictors developed recently, such as BESTORF (Solovyev and Salamov 1999a) and ESTscan (Iseli et al. 1999), significantly improve the accuracy of earlier approaches.

The internal exon prediction program SORFIND (Hutchinson and Hayden 1992) was designed based on codon usage and Berg and von Hippel (1987) discrimination energy for intron-exon boundary recognition. The accuracy of exact internal exons prediction (at both 5′- and 3′-splice junctions and in the correct reading frame) by the SORFIND program reaches 59 percent, with a specificity of 20 percent. Snyder and Stormo (1993) applied a dynamic programming approach (an alternative to the rule-based approach) to internal exon prediction in the GeneParser algorithm. It recognized 76 percent of internal exons, but the structure of only 46 percent of the exons was exactly predicted when tested using entire GenBank entry sequences. The HEXON (Human EXON) program (Solovyev et al. 1994a), based on linear discriminant analysis, was one of the most accurate in exact internal exon prediction. It was recently upgraded to predict all type of exons (and renamed FEX—find exon) (Solovyev et al. 1994b). The FEX program can be useful to analyze a possible set of alternatively spliced exons in a given sequence in addition to the optimal variant of gene structure produced by exon assembling programs. In an effort to improve the accuracy of exon prediction, Zhang (1997) applied quadratic discriminant technique (in the MZEF program) as a direct extension of the classical liner discriminant approach used in the HEXON program. The statistical evaluation of MZEF predictions on 473 genes (partially included in MZEF training) demonstrated a better performance than the HEXON program.

Later, a number of single-gene prediction programs were developed to assemble potential eukaryotic coding regions into translatable mRNA sequences, selecting optimal combinations of compatible exons (Fields and Soderlund 1990; Gelfand 1990; Guigo et al. 1992; Dong and Searls 1994). Dynamic programming was suggested as

a fast method to find an optimal combination of pre-selected exons (Gelfand and Roytberg 1993; Solovyev and Lawrence 1993b; Xu et al. 1994), which is different from the approach in the GeneParser algorithm suggested by Snyder and Stormo (1993) to recursively search for exon-intron boundary positions. The FGENEH (Find GENE in Human) algorithm incorporated 5′-, internal, and 3′-exon identification linear discriminant functions and a dynamic programming approach (Solovyev et al. 1994, 1995). Burset and Guigo (1996) conducted a comprehensive test for it and the other gene finding algorithms; the FGENEH program was one of the best in the tested group, having an exon prediction accuracy 10 percent higher than the others and the best level of accuracy on the protein level. A novel step in gene prediction approaches was the application of generalized hidden Markov models implemented in the Genie algorithm (Kulp et al. 1996). Genie is similar in design to GeneParser, but is based on a rigorous probabilistic framework. It is similar to FGENEH in performance (Kulp et al. 1996).

9.4 Multiple Gene Prediction by Discriminative and Probabilistic Approaches

Whole genome sequencing projects were initiated for a number of organisms, from bacteria to higher eukaryotes. They require gene-finding approaches that are able to identify many genes encoded in the genomic sequences. The most accurate multiple gene prediction programs include such HMM-based probabilistic approaches as Genscan (Burge and Karlin 1997) and Fgenesh (Salamov and Solovyev 2000), Fgenes (discriminative approach) (Solovyev 1997), and Genie (generalized HMM with neural network splice site detectors) (Reese et al. 2000). In the next section, we will describe a general scheme of HMM-based gene prediction that was initially realized in the works of Dr. Haussler's group (Krogh et al. 1994; Kulp et al. 1996). This pattern-based approach can also be considered as a variant in which transition probabilities are not taken into account.

9.4.1 HMM-Based Eukaryotic Gene Identification

Exons, introns, 5′-, and 3′-untranslated regions are different components (states) of gene structure that occupy k non-overlapping subsequences of a sequence $X = \bigcup_{i=1,k} x_i$. There are 35 states constituting an eukaryotic gene model, considering direct and reverse chains as possible gene locations (figure 9.9). The absence of protein coding characteristics reduces significantly prediction accuracy of noncoding 5′- and 3′-exons (and introns; therefore, they are not considered in the current gene prediction algorithms. The other 27 states consist of six exon states (first, last, single, and three types of internal exons due to three possible reading frames) and seven non-

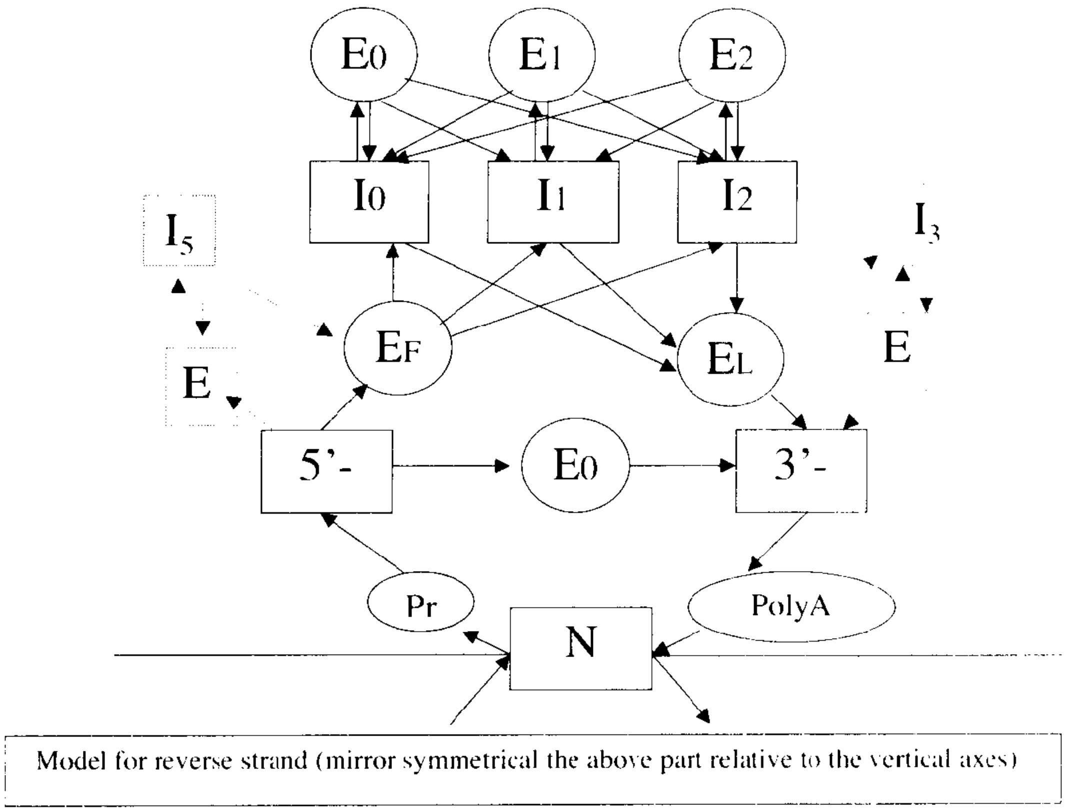


Figure 9.9
Different states and transitions in eukaryotic gene model.

coding states (three intron, noncoding 5′- and 3′-, promoter, and polyA) in each chain, plus the noncoding intergenic region.

A gene structure can be considered as the ordered set of state/subsequence pairs, $\phi = \{(q_1, x_1), (q_2, x_2), \ldots, (q_k, x_k)\}$, called the parse. We call the predicted gene structure to be such a parse ϕ that the probability $P(X, \phi)$ of generating X according to ϕ is maximal over all possible parses (or when some score is optimal in some meaningful sense, i.e., best explains the observations [Rabiner 1989]). This probability can be computed using statistical parameters describing a particular state and generated from the training set of known gene structures and sequences.

Successive states of this HMM model are generated according to the Markov process with the inclusion of explicit state duration density. A simple technique based on the dynamic programming method for finding the optimal parse (or the single best state sequence) is called the Viterbi algorithm (Forney 1973). The algorithm requires $o(N^2D^2L)$ calculations, where N is the number of states, D is the longest duration, and L is the sequence length (Rabiner and Juang 1993). Burge (1997) introduced a helpful technique to reduce the number of states and simplify computations by

modeling noncoding state length with a geometrical distribution. We will shortly consider the algorithm of gene finding using these technique, which was initially implemented in the Genscan program (Burge 1977; Burge and Karlin, 1997) and used later in the Fgenesh program (Salamov and Solovyev 2000). As any valid parse will consist of only alternating series of noncoding and coding states, NCNCNC, ..., NCN, we need only 11 variables, corresponding to the different types of N states. For each sequence position (starting from 1), we select the maximum joint probability to continue the current state or to move to another noncoding state defined by a coding state (from a pre-computed list of possible coding states) that terminates at the analyzed sequence position. The parse probability is

$$P(X, \ddot{o}) = P(q_1) \sum_{i=1}^{k-1} P(x_i \mid l(x_i), q_i) P(l(x_i) \mid q_i)(P(q_{i+1}, q_i)) P(x_i \mid l(x_k), q_k) P(l(x_k) \mid q_k)$$

where $P(q_1)$ denotes the initial state probability; $P(x_i \mid l(x_i), q_i)P(l(x_i) \mid q_i)$ and $P(q_{i+1}, q_i)$ are the independent joint probabilities of generation of the subsequence x_i of length l in the state q_i and transitioning to the q_{i+1} state. $P(x_i \mid l(x_i), q_i)P(l(x_i) \mid q_i)$ is a production of a probability of generation l-length sequence x_i and the probability of observing such an l-length sequence in the state q_i, which are computed using the sequence of x_i and the statistical data from a training set of known genes. To compute $P(x_i \mid l(x_i), q_i)$ for an internal exon state, we use donor and acceptor site models based on position specific weight matrices and frame-specific Markov models based on hexaplet frequencies in exons and introns.

Search for Optimal Parse Let us define the best score (the highest joint probability $\gamma_i[j]$ of the optimal parse of the subsequence $S_{1,j}$ $[s_1, s_2, \ldots, s_j]$, which ends in state q_i at position j). Assume a set A_j of coding states $\{c_k\}$ of lengths $\{d_k\}$, starting at positions $\{m_k\}$ and ending at position j, which have the previous states $\{b_k\}$. The length distribution of state c_k is denoted by $f_{c_k}(d)$. The searching procedure can be stated as follows:

INITIALIZATION:

$$\gamma_i(1) = \pi_i P_i(s_1) p_i \quad \text{and} \quad Z_i(1) = 0, \quad i = 1, \ldots 11$$

RECURSION:

$$\gamma_i(j+1) = \max\left\{ \gamma_i(j) p_i P_i(s_{j+1}), \max_{c_k \in A_j} \{\gamma_i(m_k - 1)(1 - p_{b_k}) t_{b_k, c_k} f_{c_k}(d_k) P(S_{m_k, j}) \times t_{c_k, i} p_i P_i(s_{j+1})\} \right\} \qquad i = 1, \ldots 11, \; j = 1, \ldots, L - 1.$$

TERMINATION:

$$\gamma_i(L+1) = \max\left\{\gamma_i(L),\ \max_{c_k \in A_i}\{\gamma_i(m_k - 1)(1 - p_{b_k})t_{b_k,c_k}f_{c_k}(d_k)P(S_{m_k,j})t_{c_k,i}\}\right\}$$

$$i = 1, \dots, 11$$

On each step we record the location and type of transition maximizing the functional to restore the optimal set of states (gene structure) by a backtracking procedure. Most parameters of these equations can be calculated from the learning set of known gene structures. Instead of scores of coding states $P(S_{m_k,j})$, it is better to use log-likelihood ratios, which do not produce scores below the limits of computer precision.

This technique to predict multiple eukaryotic genes was initially implemented in the Genscan algorithm (Burge and Karlin 1997). Several other HMM-based gene prediction programs were developed later: Veil (Hederson et al. 1997), HMMgene (Krogh 1997), Fgenesh (Salamov and Solovyev 1999, 2000), a variant of Genie (Kulp et al. 1996), and GeneMark (Lukashin and Borodovsky 1998). Fgenesh (Find GENES using Hmm) is currently the most accurate program. It is different from Genscan in computing the coding scores of potential exons, where a priori probabilities of exons were taken into account according to the Bayes theorem. As a result, the coding scores of potential exons are generally lower than in Genscan. Some minor differences exist in the functional signal description and preparing of training sets to compute specific parameters for each model organism, such as human, Drosophila, nematode, yeast, Arabidopsis, monocotolydons, and so on. Coding potentials were calculated separately for four isochores (human) and for two isochores (other species). The run time of Fgenesh is practically linear; the current version has no practical limit on the length of analyzed sequence. Prediction of about one thousand genes in 34.5 MB of chromosome 22 sequence takes about 1.5 minutes with a Dec-alpha processor EV6.

9.4.2 Discriminant Analysis–Based Gene Prediction

The Fgenes (*Find GENES*) program predicts multiple genes using dynamic programming and discriminant classifiers to generate a set of exon candidates. The following major steps describe analysis of genomic sequences by the Fgenes algorithm:

1. Create a list of potential exons by selecting: ATG ... GT, AG-GT, AG. Stop sequence regions having exons scores higher than the specific thresholds depending on GC content (four groups);

2. Find a set of compatible exons with the maximal total score. Guigo (1999) described an effective procedure for finding such a set. Fgenes uses a simpler variant of

Table 9.9
Significance of internal exon characteristics selected by LDA

	Characteristics	1	2	3	4	5
a	Individual D^2	15.0	12.1	0.4	0.2	1.5
b	Combined D^2	15.0	25.3	25.8	25.8	25.9

Characteristics 1 and 2 are the values of donor and acceptor site LD recognition functions. 3 gives the octanucleotide preference for being coding of potential exon region. 4 gives the octanucleotide preference for being intron 70-bp region on the left and 70-bp region on the right of potential exon region.

this algorithm: we order all exon candidates according to their 3′-end positions. Then, going from the first to the last exon, we select for each exon the maximal score path (compatible exons combination) terminated by this exon using a dynamic programming approach. Include in the optimal set either this exon or an exon with the same 3′-splicing pattern ending at the same position or earlier, whichever has the higher maximal score path.

3. Take into account promoter or polyA scores (if predicted) in terminal exon scores. The run time of the algorithm grows approximately linearly with the sequence length. Fgenes is based on usage linear discriminant functions developed for identification of splice sites, exons, promoter, and polyA sites (Solovyev et al. 1994; Solovyev and Salamov 1997). We consider these functions for internal exons to demonstrate what sequence features are important to exon identification.

Internal Exon Recognition We consider as potential internal exons all open reading frames in a given sequence flanked by AG (on the left) and GT (on the right). The structure of such exons is presented in figure 9.10. The values of five exon characteristics were calculated for 952 authentic exons and for 690,714 pseudo-exon training sequences from the set. Table 9.9 gives the Mahalonobis distances, showing the significance of each characteristic. We can see that the strongest characteristics are the recognition functions of flanking donor and acceptor splice sites ($D^2 = 15.04$ and $D^2 = 12.06$, respectively). The preference of ORF as a coding region has $D^2 = 1.47$ and adjacent left intron region has $D^2 = 0.41$ and right intron region has $D^2 = 0.18$.

The accuracy of the discriminant function based on these characteristics was estimated on the recognition of 451 exon and 246,693 pseudo-exon sequences from the test set. The sensitivity of exact internal exon prediction is 77 percent, with a specificity of 79 percent. At the level of individual nucleotides, the sensitivity of exon prediction is 89 percent, with a specificity of 89 percent; the sensitivity of intron positions prediction is 98 percent, with a specificity of 98 percent. This accuracy is better than that demonstrated by dynamic programming and neural network based methods

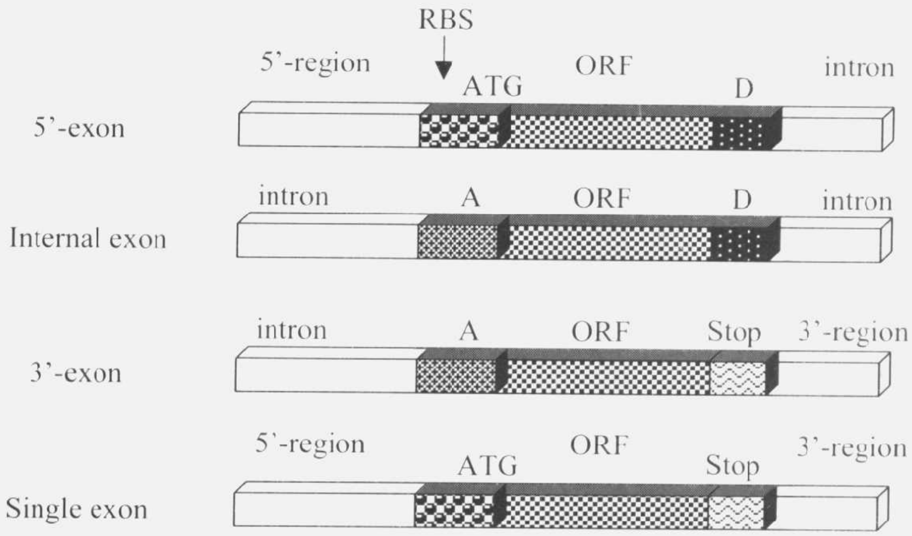


Figure 9.10
Different functional regions of the first (a), internal (b), last (c) corresponding to components of recognition functions. Single exons include left and right characteristics of first and last exons, respectively.

(Snyder and Stormo 1993), which have a 75 percent sensitivity of the exact internal exons prediction, with a specificity of 67 percent.

5′_coding exon recognition LDF uses the average value of positional triplet preference in the (_15, 10) region around ATG codon (instead of donor splice site score). 3′-exon coding region recognition LDF includes the average value of positional triplet preference in the (_10, 30) region around the stop codon (instead of the acceptor site score). The recognition function of single exons combines corresponding characteristics of 5′- and 3′-exons (figure 9.10) (Solovyev et al. 1994; Solovyev and Salamov 1997). Features describing sequences near initial and stop codons have much less discriminative power than the splice site characteristics; therefore, terminal and short single exons have a lower accuracy of recognition.

9.5 Accuracy of Gene Identification Programs

Burset and Guigo (1996) specially selected a set of 570 single-gene sequences of mammalian genes, which they used to evaluate the performance of many gene finding approaches. The results of this test are presented in table 9.10. Of course, some of these data have only a historical value to show the progress in gene finding development, and some of these programs have been improved since the test. We can see that the best programs on average predict accurately 93 percent of exon nucleotides

Table 9.10
Characteristics of accuracy for the gene prediction programs on single gene sequences of Burset and Guigo 1996 dataset

Algorithm/ program	Sn (exons)	Sp (exons)	Sn nucleo-tides	Sp nucleo-tides	Reference
Fgenesh	0.84	0.86	0.94	0.95	Salamov & Solovyev 1998
Fgenes	0.83	0.82	0.93	0.93	Solovyev 1997
Genscan	0.78	0.81	0.93	0.93	Burge & Karlin 1997
Fgeneh	0.61	0.64	0.77	0.88	Solovyev et al. 1995
Morgan	0.58	0.51	0.83	0.79	Salsberg et al. 1998
Veil	0.53	0.49	0.83	0.79	Henderson et al. 1997
Genie	0.55	0.48	0.76	0.77	Kulp et al. 1996
GenLang	0.51	0.52	0.72	0.79	Dong & Searls 1994
Sorfind	0.42	0.47	0.71	0.85	Hutchinson & Hyden 1992
GeneID	0.44	0.46	0.63	0.81	Guigo et al. 1992
Grail2	0.36	0.43	0.72	0.87	Xu et al. 1994
GeneParser2	0.35	0.40	0.66	0.79	Snyder & Stormo 1995
Xpound	0.15	0.18	0.61	0.87	Thomas & Skolnick 1994

Sn (sensitivity) = number of exactly predicted exons/number of true exons (or nucleotide); Sp (specificity) = number of exactly predicted exons/number of all predicted exons. Accuracy data for programs developed before 1996 were estimated by Burset and Guigo (1996). The other data were produced by the authors of the corresponding programs.

(Sn = 0.93), with just 7 percent false positive predictions. However, the accuracy on the nucleotide level does not completely reflect the quality of gene structure prediction because missing small exons and the imperfect location of exon ends will not much affect its value. Therefore, it is important to provide the accuracy of exact exon prediction level, which is usually lower than at the nucleotide level.

The table clearly demonstrates that the recent multiple gene prediction programs such as Fgenesh, Fgenes, and Genscan significantly outperform the older approaches. The exon identification rate is actually even higher than the presented data because the overlapped exons were not counted as true predictions in exact exon accuracy evaluation. Yet there is still room for significant improvement. The accuracy of exact gene prediction is only 59 percent for Fgenesh, 56 percent for Fgenes, and 45 percent for Genscan programs computed on this relatively simple test with single gene sequences.

A more practical task is to identify multiple genes in long genomic sequences containing genes in both DNA strands. We selected a test set of 19 long genomic sequences of 26,000–240,000 bp and 19 multigene sequences with 2–6 genes from GenBank to compare performance of gene-finding programs in analyzing genomic DNA. Table 9.11 demonstrates the results of gene prediction for these data. The results show that

Table 9.11
Performance of gene-finding for 38 genomic sequences

Program	Sequences/ Genes	Accuracy per nucleotide			Accuracy per exon				Genes/ Entries
		Sn	Sp	CC	Sn/Sn_o	Sp	Me	We	
Fgenesh	38/77 M_r	0.94	0.87	0.90	0.85/0.93	0.80	0.08	0.14	0.36/0.11
		0.94	0.78	0.85	0.84/0.92	0.75	0.08	0.21	0.34/0.08
Genscan	38/77 M_r	0.93	0.82	0.87	0.80/0.90	0.74	0.10	0.18	0.29/0.03
		0.92	0.70	0.79	0.79/0.90	0.66	0.11	0.30	0.29/0.03
Fgenes	38/77 M_r	0.91	0.80	0.84	0.84/0.92	0.72	0.08	0.21	0.36/0.18
		0.92	0.76	0.83	0.84/0.93	0.68	0.07	0.30	0.39/0.21

Me, missing exons; WE, wrong exons. M_r lines provide predictions on sequences with masked repeats. Sn_o, exon prediction accuracy including overlapping exons.

the accuracy is still pretty good on the nucleotide and exon level, but exact gene prediction is lower than for the test with short single gene sequences. Sensitivity for exact internal exon prediction is 85–90 percent, but 5′-, 3′-, and single exons have a prediction sensitivity of about 50–75 percent, which can partially explain relatively low level of exact gene prediction. As a result, we observe the splitting of some actual genes and/or joining some other multiple genes into a single one.

Another limitation of current gene-finding programs is that they cannot detect the nested genes, that is, genes located inside introns of other genes. This is one of the future directions for improvement of gene-finding software. Although this is probably a rare event for the human genome, for organisms like Drosophila, it presents a real problem. For example, annotators identified 17 examples of such cases in the Adh region. (Ashburner et al. 1999). Masking repeats is important. It significantly increases (~10 percent) the specificity of prediction.

9.6 Knowledge of Similar Protein or EST Can Improve Gene Prediction

Automatic gene prediction approaches can take into account some information about exon similarity with a known protein or EST (Gelfand et al. 1996; Xu and Uberbacher 1996; Krogh 2000; Birney and Durbin 2000). Fgenesh+ (Salamov and Solovyev 2000) is a modification of the Fgenesh algorithm, which uses additional information from available similar proteins. These proteins can be acquired by running Fgenesh on a given sequence. Then the predicted proteins (or amino acid fragments translated from predicted exons) are used to select similar proteins in some protein database. After that, we can use selected proteins to improve prediction accuracy. Fgenesh+

Table 9.12
The accuracy of Fgenesh and Fgenesh+ on the same set of human genes with known protein homologs from another organisms

Program	CG	Sn_e	Sp_e	Sn	Sn	CC
Fgenesh	0	63	68	86	83	0.74
Fgenesh+	46	82	85	96	98	0.95

The set includes 61 genes and 370 exons. CG percent of correctly predicted genes; Sn_e, Sp_e sensitivity and specificity at the exon level (in %); Sn, Sp sensitivity and specificity at the nucleotide level (in %); CC correlation coefficient.

reads the protein homolog sequence and aligns all predicted potential exons with that protein using the Smith-Waterman algorithm, as implemented in the *Sim* program (Huang et al. 1990) or the *Lial* (local iterative alignments) algorithm developed by Seledtsov and Solovyev (1999). To improve the computational time, all overlapped exons in the same reading frame are combined into one sequence and aligned only once.

Fgenesh+ includes two major additions to the Fgenesh algorithm: augmentation of the exon score (for exons having detected similarity) by an additional term proportional to the alignment score and imposing a penalty for the adjacent exons in a dynamic programming procedure, when the distance between their corresponding protein segments is significantly different from the distance between the corresponding fragments of a similar protein. We tested Fgenesh+ on a set of 61 GenBank human sequences, which have imperfect ab initio Fgenesh predictions and known protein homologs from other organisms with identity varying from 99 percent to 40 percent. The results of applying Fgenesh+ to these sequences show (table 9.12) that when the alignment covers the entire lengths of both proteins, the accuracy increases (relative to Fgenesh) and the improvement does not depend significantly on the level of percent identity (for ID > 40 percent). This feature makes valuable the proteins from distant organisms for improving the accuracy of gene identification. Having a sequence of the human genome, we can find where in the genomic sequence a given protein is located using Blast-like search in all predicted proteins of this genome. Then we use Fgenesh+ for a prediction of the full-length mRNA (its coding part) for a given protein using its sequence and the selected genomic sequence. Recently we have developed a Fgenesh++ script, which initially predicts genes using the Fgenesh program and then selects from NR (nonredundant protein database) similar proteins for predicted genes using the Dbscan program (a Blast-like program, but about 10 times faster). The Fgenesh++ script uses found protein sequences to improve initial gene prediction and can automatically generate annotation of the entire chromosome.

Similar to the Fgenesh+ algorithm scheme of exploiting known EST/cDNA information to improve accuracy of gene identification is the program Fgenesh_c (Salamov and Solovyev 2000a). Fgenesh+ and Fgenesh_c are very fast programs. For example, gene prediction by Fgenesh+ for a sequence of 80,000 bp with a protein of eight hundred amino acids takes about 1 second on an EV6 processor in a Dec-alpha computer.

9.7 Annotation of Genomic Sequences

GenBank (Benson et al. 1999) and EMBL (Stoesser et al. 1999) databases have for many years collected information about sequences of different genomes. A sequence-based structure of these databases often produces annotations of one gene in many different records when several gene fragments are sequenced independently. Last year, the vast amount of sequence information was produced by genome sequencing projects. Absence of experimental information about genes in a major part of these sequences makes valuable a presentation of computationally identified genes to provide positional cloners, gene hunters, and others with the gene candidates contained in finished and unfinished genomic sequences. Using these predictions, the scientific community can experimentally work with most real genes, because gene finding programs usually predict correctly most exons in a gene sequence.

9.7.1 Gene-Centered InfoGene Database

The *InfoGene* database database (http://www.softberry.com/inf/infodb.html) is created to collect and interactively work with information about known and predicted gene structures of human and other model genomes. Known genes are presented in 17 separate divisions (including human, mouse, *Drosophila*, nematode, *Arabidopsis*, rice, maize, and wheat), which contain records uniting available information about a particular gene from many GenBank (Release 119) entries. The human *InfoGene* division, for example, contains about 20,791 genes (including 16,141 partially sequenced genes), 54,558 coding regions, 83,488 exons, and about 58,000 donor and acceptor splice sites. This information can be applied to create different sets of functional gene components for extraction of their significant characteristics as used in gene prediction systems.

The interactive Java Viewer of Gene Structures has been designed by Igor Seledtsov and Victor Solovyev (1999) to visually inspect the gene structure of Infogene entries of known genes and predicted genes and to use for analysis of different gene prediction algorithms in annotating genomic sequences from genome sequencing projects.

The viewer has four main panels (see figures 9.2, 9.12): General View, Detail View, Locus Selection Panel, and the Output Message Panel. Both the General View Panel and the Detail View Panel have horizontal zoom scroll bars at the bottom of the windows. You can also zoom in and out by entering a scaling value from a pull-down menu: Action > Set Horizontal Scaling Factor.

9.7.2 General View Panel

The General View Panel shows all genes found in a given locus. If the locus contains overlapping genes, such as alternatively spliced ones, every such gene is displayed on its own line. As an example, let us look at the InfoGene locus HSAB001898. This locus should be automatically displayed in the General View Window if you have chosen the default setting from the Infogene page (the button "Show data" on this page). This example is also shown on the picture above. Genes are shown as red bars. When the mouse cursor points to one of the genes, the gene's name is displayed in the Output Message Panel. If you press and hold the right mouse button, detailed information about the gene is displayed in a separate temporary window. When the button is released, the temporary window disappears. The same operation performed with the Shift key leaves the temporary window open after the release of the mouse button. The number of temporary information windows that can be opened at the same time is unlimited. You can mark genes or groups of genes in General View Panel by pressing and dragging the left mouse button. Marked gene(s) will then be displayed in the Detail View Window, replacing its previous content. If you press the Shift key at the same time as pressing and dragging the left mouse button, you can add new marked regions to the current ones. Then all marked regions will be displayed one under another in the Detail View Panel.

9.7.3 Detail View Panel

This panel offers detailed view of selected genes. Each selected gene or region is drawn on a separate line, the number of which is unlimited. Dark gray bars represent genes, yellow bars show exons, red bars show coding regions, and green bars show gene regions that are not included in a transcript. Symbols < and > at the end of exon or coding region mean that their exact boundaries are unknown. White separators that cut through genes separate the unsequenced regions. Colored triangles above genes represent functional signals: black—CAAT box, blue—TATA box, green—transcription start point, red—PolyA signal, pink—polyA site. When the mouse cursor points to any bar or triangle, information about this object is displayed in the Output Message Panel. If the mouse cursor goes across a gene, the gene name is displayed in the Output Message Panel.

9.7.4 Locus Selection Panel

This panel has five fields: Selection List, Locus Info Button, Back Button, Forward Button, and Search Input Window. Selection List shows the list of sequence identifiers that satisfy current criteria, set forth in the Divisions, Options, and Search Fields menus of the Viewer. The Divisions menu allows the user to choose the source of data: Genbank/Infogen known genes for several taxonomic group, or predicted genes for several organisms. The Options menu allows the user to chose a field to search in Infogen loci, GenBank Identifiers, GenBank Accession Codes, or Context. The latter option allows a search through all fields, performed by typing a search string (the wildcard * is allowed) into the Search Input Window and pressing Enter. Entries that satisfy search criteria are displayed in a Selection List. To display an entry from the list in the General View panel, double-click on an entry or select it and press the Enter key. The Forward and Back buttons display the next/previous thousand entries in selection list. Pressing the Locus Info button opens a separate window with detailed information on a given locus.

InfoGene exon-intron gene structures can be visualized by Dbscan program (http://www.softberry.com/scan.html), which searches for conserved regions in two sequences. This tool is useful to compare the localization of conservative regions with the localization of corresponding exon sequences or gene regulatory signals. In figure 9.11, we present the results of searching similar regions for mouse mRNA in a database of known human genes. We can see that all exons shown by red boxes (in the second window) have corresponding conserved regions (in the first window). In this way we can see exon boundaries in the mouse RNA. If we use predicted gene database and a given mRNA sequence, we can verify the corresponding predicted exons using this tool.

9.7.5 Predicted Genes in the *Drosophila* Genome

The Predicted Genes division includes an annotation of a draft of the *Drosophila* genome and a draft human genome sequence. The nucleotide sequence of nearly all euchromatic portion of the *Drosophila* genome (~120 MB) has been determined (Adams et al. 2000). These sequences were annotated by predicting genes with the Fgenesh program and checking exon similarity with PfamA domains (Bateman et al. 2000). The results of this analysis are shown in table 9.13. In this table, in addition to computer predicted genes, also shows the results of removing (filtering out) most unreliable genes. Two criteria were used: (1) remove genes with the total length of protein coding region less than 30 amino acids; and (2) remove genes with total score of exons < 15. Such filtering proved useful to improve the accuracy of gene predic-

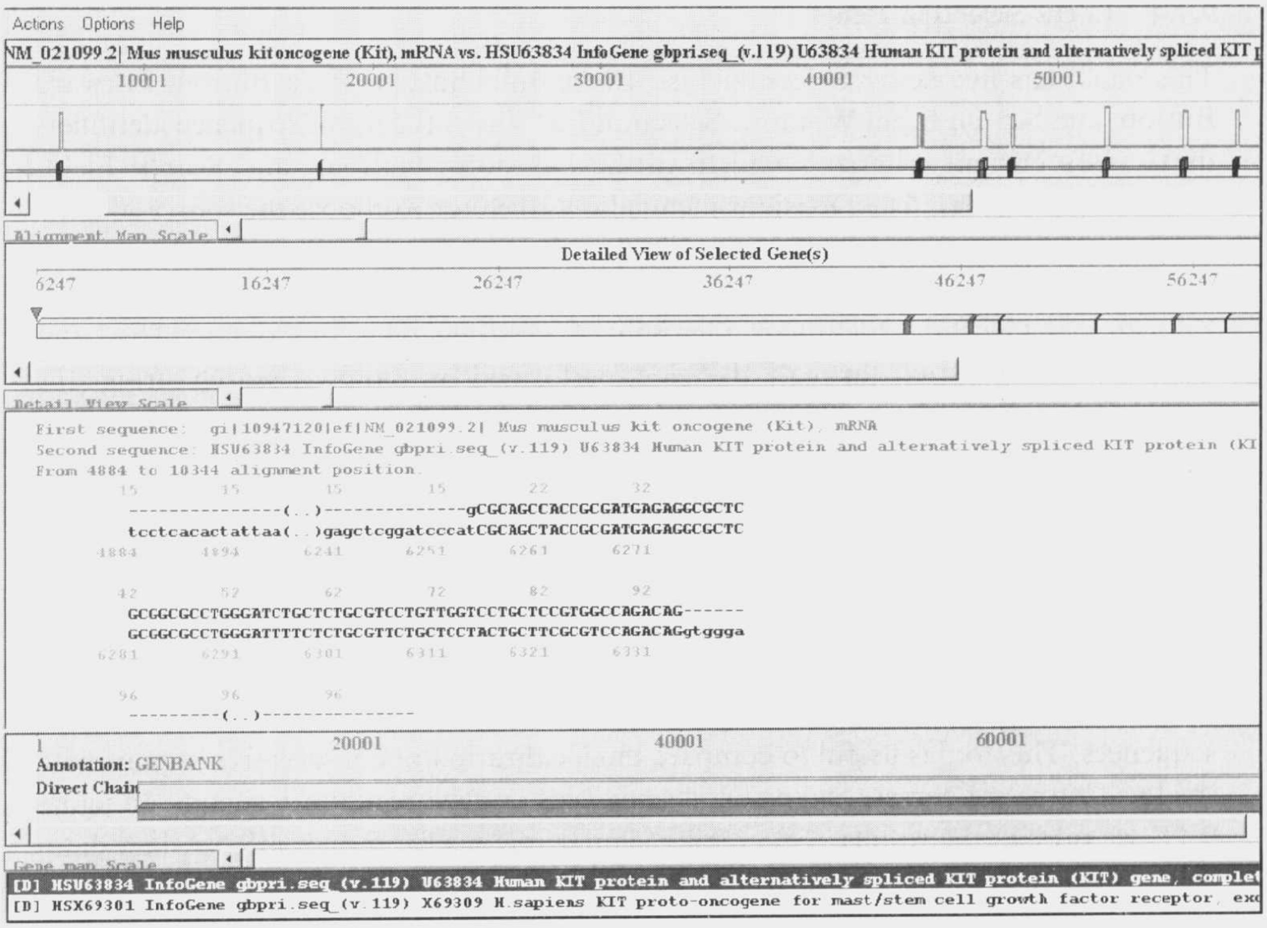


Figure 9.11
Dbscan visualization of results of searching similar regions for a mouse mRNA in a database of known human genes. We can see that all exons shown by red boxes (in the second window) have corresponding conserved regions (in the first window). By this we can detect exon boundaries in the mouse RNA. In third window we can display alignment by marking some conserved region in the first window.

Table 9.13
Summary of predicted genes and proteins in *Drosophila* genome sequences

	X	2L	2R	3L	3R	4	Y	Unknown	Total
Size (MB)	22.2	23.0	21.4	24.1	28.3	1.2	0.02	4.6	124.8
All genes	4071	4610	4573	4851	5954	133	1	691	24884
filtered	3349	3768	3915	4017	4962	105	1	504	20622
All exons	13036	15215	16310	16047	20382	679	10	1804	83483
filtered	11767	13713	15138	14561	18654	625	10	1467	75935
Exons-PfamA	1932	2148	2348	2130	2919	109	0	159	11745
filtered	1925	2141	2341	2126	2916	105	0	147	11701
United Pfam	1138	1193	1287	1216	1654	58	0	76	6622
Pfam types	431	475	499	460	546	43	0	40	1017

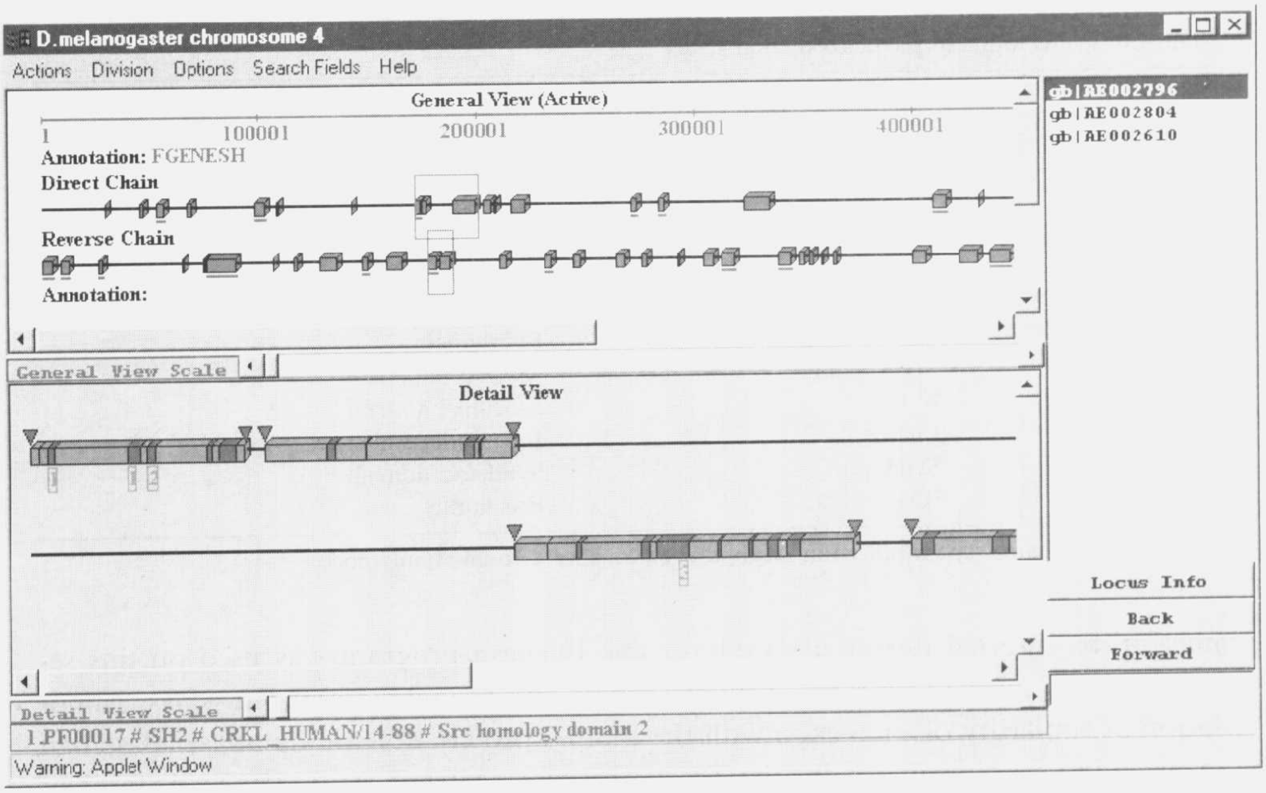


Figure 9.12
InfoGene viewer representation of Fgenesh annotation of chromosome 4 of *Drosophila melanogaster*. Genes marked in the upper panel are presented in the lower panel. Coding exons are marked by red and introns by dark color. The triangles show the starts of transcription and the poly-A signals. Underlined red genes have similarity with Pfam domains. Pointing with the mouse to the first exon, we can see in down information line the similarity with Src domain; the exon in reverse chain (marked by 3) has a similarity with EGF-like domain.

tion (Salamov and Solovyev 2000). We should note that 20,622 genes include some pseudogenes and genes of mobile elements. The sequences of exons and gene annotation data can be copied from http://www.softberry.com/inf/dro_ann.html for using them locally or to create microarray oligos.

The predicted genes and proteins for each human chromosome can be seen in figure 9.12, and used for further investigation at http://www.softberry.com/inf/infodb.html.

9.7.6 Predicted Genes in the Human Genome

The nucleotide sequence of nearly 90 percent of the human genome (3 GB) has been determined by an international sequencing effort. Assembly of the current draft of the human genome was done by Prof. Haussler's Human Genome Project Team at UC Santa Cruz. Half of this sequence is occupied by repeat sequences and undefined

Table 9.14
PfamA domains found in predicted human genes

Number	PfamA short name	Name
467	Pkinase	Eukaryotic protein kinase domain
372	7tm_1	7 transmembrane receptor (rhodopsin family)
308	Myc_N_term	Myc amino-terminal region
256	Topoisomerase_I	Eukaryotic DNA topoisomerase I
224	Ig	Immunoglobulin domain
183	Rrm	RNA recognition motif
182	PH	PH domain
180	Myosin_tail	Myosin tail
166	EGF	EGF-like domain
159	Filament	Intermediate filament proteins
154	Syndecan	Syndecan domain
143	Ras	Ras family

Domain of the same type localized in neighboring exons were counted only once.

nucleotides inserted during assembling. The Fgenesh program was used on this sequence (with masked repeats) to predict exons and assemble predicted genes. Annotation of similarity of each exon with the PfamA protein domain database (Bateman et al. 2000) was produced by the Blast program (Altshul et al. 1977). A total of 49,171 genes and 282,378 coding exons were predicted. On average, one gene was found per about 68,623 bp, and one exon per 11,949 bp. Complete summary of this analysis including the gene and exon numbers in different chromosomes. is presented at http://www.softberry.com/inf/humd_an.html and can be viewed in the InfoGene database. Sequences of predicted exons and gene annotation data can also be copied from this site. One thousand one hundred and fifty-four types of PfamA different domains were found in the predicted proteins. The top part of the domain list is presented in table 9.14.

9.8 Using Expression Data for Characterization and Verification of Predicted Genes

Large-scale functional analysis of predicted, as well as known, genes might be done using expression micro array technology, which gives us the possibility of presenting all human genes on one or several Affymetrix type GeneChips. Traditionally genes are presented on the chips by unique oligonucleotides close to the 3′-end of the mRNA, but there are a lot of predicted new genes that have no known corresponding EST sequences. However, the expression of such genes could be studied using predicted exon sequences. We can present all predicted human exons (about 300,000) on

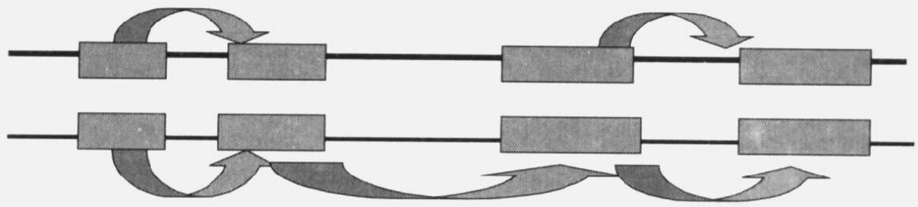


Figure 9.13
Using expression data to verify gene boundaries and reality of exons predicted along genomic sequences. Real exons will have expression signals. Exons from one gene will have the same expression pattern in different tissues. Expression signal will increase toward 3′-exons.

a few chips and use expression profiling across many tissues to verify the predicted exons, observing if they are expressed in some of them.

Moreover, with this approach, we can verify the structure of genes (identify a subset of predicted exons that really belong to the same gene) based on the similar expression behavior of exons from the same gene in a set of tested tissues. Exons wrongly included in a predicted gene will have different expression patterns, and exons wrongly excluded by prediction will have similar expression patterns (figure 9.13). It is interesting that such gene verification on a large scale can be done in parallel with identification of disease (tissue) specific drug target candidates. The recent chip designed by EOS Biotechnology included all predicted by Fgenesh and Genescan exons from chromosome 22, as well as predicted exons from human genomic sequences of phase 2 and 3. It was found that the predicted exon sequences present a good alternative to EST sequences, which opens a possibility of working with predicted genes on a large scale.

An example of expression behavior of three exons of the myoglobin gene in different tissues is presented in figure 9.14 (expression data were received in EOS Biotechnology Inc.). Tissue-specific expression of this gene is clearly seen with the major peaks located in skeletal muscle, heart, and diaphragm tissues. The level of expression in these tissues is 10–100 times higher than the level of signals for other tissues, as well as the average level of expression signal for randomly chosen exons. We can observe that for specific tissues, all three exons demonstrate such a high level (with correlation coefficient 0.99; for random exons it is about 0.06). These exons were predicted correctly by the Fgenesh program and were used for selection of oligonucleotide probabilities. From this result we can conclude that the predicted exons can be used as a gene representatives. An additional application of expression data is the

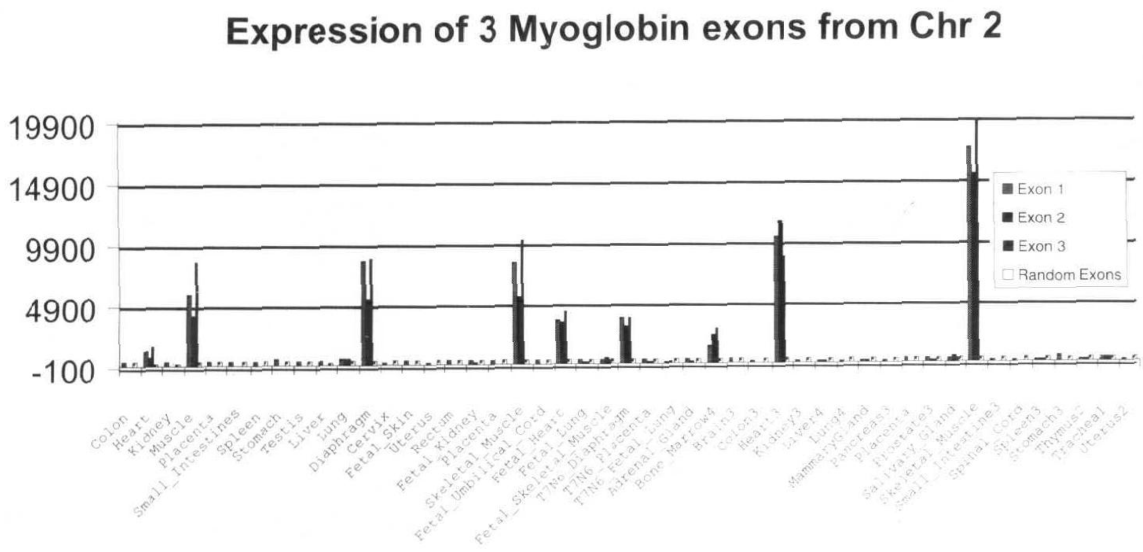


Figure 9.14
Coordinative expression of three exons of human myoglobin gene from chromosome 22 (exons were predicted by Fgenesh program and used to design EOS Biotechnology Human genome chip). The high level of expression is observed only in several specific tissues.

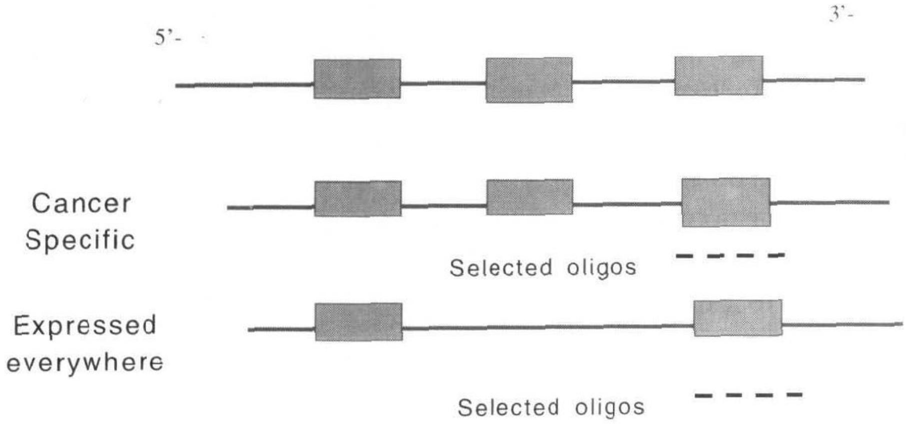


Figure 9.15
Exon representation can be used to characterize alternatively spliced variants of genes. Oligonucleotides selected in 3′-end of mRNA/EST sequences will not be selective for different gene variants. We will observe the sum of two genes' signal and we can miss cancer specificity of three exons' gene structure. Using oligonucleotides derived from exon sequences, we can detect different expressions of these two forms.

Table 9.15
Web servers for eukaryotic gene and functional signal prediction

Program/task	WWW address
Fgenesh/HMM-based gene prediction (human, *Drosophila*, dicots, monocots, *C. elegans*, *S. pombe*)	http://genomic.sanger.ac.uk/gf/gf.shtml http://searchlauncher.bcm.tmc.edu:9331/seq-search/gene-search.html http://www.softberry.com/nucleo.html
Genscan/HMM-based gene prediction (human, *Arabidipsis*, maize)	http://genes.mit.edu/GENSCAN.html
HMM-gene/HMM-based gene prediction (human, *C. elegans*)	http://www.cbs.dtu.dk/services/HMMgene/
Fgenes/Disciminative gene prediction (human)	http://genomic.sanger.ac.uk/gf/gf.shtml http://searchlauncher.bcm.tmc.edu:9331/seq-search/gene-search.html
Fgenes-M/Prediction of alternative gene structures (human)	http://genomic.sanger.ac.uk/gf/gf.shtml http://www.softberry.com/nucleo.html
Fgenesh+/Fgenesh_c/gene prediction with the help of similar protein/EST	http://www.softberry.com/nucleo.html
Fgenesh-2/gene prediction using 2 sequences of close species	http://genomic.sanger.ac.uk/gf/gf.shtml http://www.softberry.com/nucleo.html
BESTORF/Finding best CDS/ORF in EST (human, plants, *Drosophila*)	http://genomic.sanger.ac.uk/gf/gf.shtml http://www.softberry.com/nucleo.html
Mzef/internal exon prediction (human, mouse, *Arabidopsis*)	http://argon.cshl.org/genefinder/
TSSW/TSSG/promoter prediction	http://searchlauncher.bcm.tmc.edu:9331/seq-search/gene-search.html http://genomic.sanger.ac.uk/gf/gf.shtml
Promoter 2.0/promoter prediction	http://www.cbs.dtu.dk/services/Promoter/
CorePromoter/promoter prediction	http://argon.cshl.org/genefinder/CPROMOTER/index.htm
SPL/splice site prediction (human, *Drosophila*, plants, yeast)	http://genomic.sanger.ac.uk/gf/gf.shtml http://www.softberry.com/nucleo.html
NetGene2/NetPGene/splice site prediction (human, *C. elegans*, plants)	http://www.cbs.dtu.dk/services/NetPGene/
Dbscan/searching for similarity in genomic sequences and its visualization altogether with known gene structure	http://www.softberry.com/nucleo.html

functional analysis and identification of alternatively spliced genes (exons), when in particular tissues some exons (or their parts) have very different expression intensities compared to the other exons from the same gene. Moreover, sometimes 3′-EST generated probabilities cannot be used for the identification of disease-specific gene variants in contrast with the using exon representation of a gene (figure 9.15).

9.9 Internet Resources for Gene Finding and Functional Site Prediction

Prediction of genes, ORF, promoter, and splice sites finding by the methods described above is available on the World Wide Web. Table 9.15 presents just a few useful programs. It does not provide a comprehensive list.

Acknowledgements

I am grateful to Asaf Salamov and Igor Seledtsov for their collaboration in the development of gene-finding and other algorithms discussed here.

References

Adams, M. D. et al. (2000). The genome sequence of Drosophila melanogaster. *Science* 287: 2185–2195.

Aebi, M., and Weissmann, C. (1987). Precision and orderliness in splicing. *Trends Genet.* 3: 102–107.

Afifi, A. A., and Azen, S. P. (1979). *Statistical Analysis: A Computer Oriented Approach*. New York: Academic Press.

Altschul, S. F., Madden, T. L., Schaffer, A. A., Zhang, J., Zhang, Z., Miller W., and Lipman D. J. (1997). Gapped BLAST and PSI-BLAST: A new generation of protein database search programs. *Nucl. Acids Res.* 25(17): 3389–3402.

Ashburner, M., Misra, S., Roote, J., Lewis, S. E., Blazej, R., Davis, T., Doyle, C., Galle, R., George, R., Harris, N., Hartzell, G., Harvey, D., Hong, L., Houston, K., Hoskins, R., Johnson, G., Martin, C., Moshrefi, A., Palazzolo, M., Reese, M. G., Spradling, A., Tsang, G., Wan, K., Whitelaw, K., Celniker, S., and Rubin, G. M. (1999). An exploration of the sequence of a 2.9-mb region of the genome of *Drosophila melanogaster*: The Adh region. *Genetics* 153(1): 179–219.

Audic, S., and Claverie, J. (1997). Detection of eukaryotic promoters using Markov transition matrices. *Comput. Chem.* 21: 223–227.

Bateman, A., Birney, E., Durbin, R., Eddy, S., Howe, K., and Sonnhammer, E. (2000). The Pfam protein families database. *Nucl. Acids Res.* 28: 263–266.

Beck, T. W., Huleihel, M., Gunnell, M., Bonner, T. I., and Rapp, U. R. (1987). The complete coding sequence of the human A-raf-1 oncogene and transforming activity of a human A-raf carrying retrovirus. *Nucl. Acids Res.* 15(2): 595–609.

Benson, D. A., Boguski, M. S., Lipman, D. J., Ostell, J., Ouellette, B. F., Rapp, B. A., and Wheeler, D. L. (1999). GenBank. *Nucl. Acids Res.* 27(1): 12–17.

Berg, O. G., and von Hippel, P. H. (1987). Selection of DNA binding sites by regulatory proteins. *J. Mol. Biol.* 193: 723–750.

Birney, E., and Durbin, R. (2000). Using GeneWise in the Drosophila annotation experiment. *Genome Res.* 10: 547–548.

Boguski, M. S., Lowe, T. M., and Tolstoshev, C. M. (1993). dbEST—database for "expressed sequence tags," *Nat. Genet.* 4(4): 332–333.

Borodovsky, M., and McIninch, J. (1993). GENMARK: Parallel gene recognition for both DNA strands. *Comp. Chem.* 17: 123–133.

Borodovsky, M., Sprizhitskii, Yu., Golovanov, E., and Alexandrov, N. (1986). Statistical patterns in the primary structures of functional regions of the genome in *Escherichia coli*. II. Nonuniform Markov models. *Molek. Biol.* 20: 1114–1123.

Breathnach, R., Benoist, C., O'Hare, K., Gannon, F., and Chambon, P. (1978). Ovalbumin gene: Evidence for a leader sequence in mRNA and DNA sequences at the exon-intron boundaries. *Proc. Natl. Acad. Sci. USA* 75(10): 4853–4857.

Breathnach, R., and Chambon, P. (1981). Organization and expression eukaryotic of split genes for coding proteins. *Annu. Rev. Biochem* 50: 349–393.

Brunak, S., Engelbreht, J., and Knudsen, S. (1991). Prediction of Human mRNA donor and acceptor sites from the DNA sequence. *J. Mol. Biol.* 220: 49–65.

Bucher, P. (1990). Weight matrix descriptions of four eukaryotic RNA polymerase II promoter elements derived from 502 unrelated promoter sequences. *J. Mol. Biol.* 212: 563–578.

Bucher, P., and Trifonov, E. (1986). Compilation and analysis of eukaryotic POLII promoter sequences. *Nucl. Acids Res.* 14: 10009–10026.

Burge, C. (1997). Identification of genes in human genomic DNA. Ph.D. dissertation, Stanford University.

Burge, C., and Karlin, S. (1997). Prediction of complete gene structures in human genomic DNA. *Journal of Molecular Biology* 268: 78–94.

Burge, C. B., Padgett, R. A., and Sharp, P. A. (1998). Evolutionary fates and origins of U12-type introns. *Molecular Cell* 2(6): 773–785.

Burset, M., and Guigo, R. (1996). Evaluation of gene structure prediction programs. *Genomics* 34(3): 353–367.

Burset, M., Seledtsov, I., and Solovyev, V. (2000). Analysis of canonical and non-canonical splice sites in mammalian genomes. *Nucl. Acids Res.* 28: 4364–4375.

Burset, M., Seledtsov, I., and Solovyev V. (2001). SpliceDB: Database of canonical and noncanonical mammalian splice sites. *Nucl. Acids Res.* (in press).

Claverie, J., and Bougueleret, L. (1986). Heuristic informational analysis of sequences. *Nucleic Acids Res.* 10: 179–196.

Claverie, J., Sauvaget, I., and Bougueleret, L. (1990). K-tuple frequency analysis: From intron/exon discrimination to T-cell epitope mapping. *Methods Enzymol.* 183: 237–252.

Collins, F., Patrinos, A., Jordan, E., Chakravarti, A., Gesteland, A., and Walters, L. (1998). New goals for the U.S. Human Genome Project: 1998–2003. *Science* 282: 682–689.

Decker, C. J., and Parker, R. (1995). Diversity of cytopasmatic functions for the 3′-untranslated region of eukaryotic transcripts. *Curr. Opin. Cell Biol. 1995* 7: 386–392.

Dietrich, R., Incorvaia, R., and Padgett, R. (1997). Terminal intron dinucleotides sequences do not distinguish between U2- and U12-dependent introns. *Molecular Cell.* 1: 151–160.

Dong, S., and Searls, D. (1994). Gene structure prediction by linguistic methods. *Genomics* 23: 540–551.

Duda, R., and Hart, P. (1973). Pattern Classification and Scene Analysis. New York: John Wiley & Sons.

Dunham, I., Shimizu, N., Roe, B., et al. (1999). The DNA sequence of human chromosome 22. *Nature* 402: 489–495.

Farber, R., Lapedes, A., and Sirotkin, K. (1992). Determination of eukaryotic protein coding regions using neural networks and information theory. *J. Mol. Biol.* 226: 471–479.

Feng, G. H., Bailin, T., Oh, J., and Spritz, R. A. (1997). Mouse pale ear (ep) is homologous to human Hermansky-Pudlak syndrome and contains a rare 'AT-AC' intron. *Hum. Mol. Genet.* 6(5): 793–797.

Fickett, J., and Hatzigeorgiou, A. (1997). Eukaryotic promoter recognition. *Genome Research* 7: 861–878.

Fickett, J. W., and Tung, C. S. (1992). Assesment of protein coding measures. *Nucl. Acids Res.* 20: 6441–6450.

Fields, C., and Soderlund, C. (1990). GM: A practical tool for automating DNA sequence analysis. *CABIOS* 6: 263–270.

Forney, G. D. (1973). The Viterbi algorithm. *Proc. IEEE* 61: 268–278.

Gelfand, M. (1989). Statistical analysis of mammalian pre-mRNA splicing sites. *Nucl. Acids Res.* 17: 6369–6382.

Gelfand, M. (1990). Global methods for the computer prediction of protein-coding regions in nucleotide sequences. *Biotechnology Software* 7: 3–11.

Gelfand, M., and Roytberg, M. (1993). Prediction of the exon-intron structure by a dynamic programming approach. *BioSystems* 30(1–3): 173–182.

Gelfand, M., Mironov, A., and Pevzner, P. (1996). Gene recognition via spliced sequence alignment. *Proc. Natl. Acad. Sci. USA.* 93: 9061–9066.

Green, M. R. (1991). Biochemical mechanisms of constitutive and regulared pre-mRNA splicing. *Annu. Rev. Cell Biol.* 7: 559–599.

Green, P., and Hillier, L. (1998). GENEFINDER, unpublished software.

Guigo, R. (1999). DNA composition, codon usage and exon prediction. In *Genetics Databases*, Bishop, M. J. (ed.), 54–80. London: Academic Press.

Guigo, R., Knudsen, S., Drake, N., and Smith, T. (1992). Prediction of gene structure. *J. Mol. Biol.* 226: 141–157.

Guigo, R. (1998). Assembling genes from predicted exons in linear time with dynamic programming. *J. Comput. Biol.* 5: 681–702.

Hall, S. L., and Padgett, R. A. (1994). Conserved sequences in a class of rare eukaryotic nuclear introns with non-consensus splice sites. *J. Mol. Biol.* 239(3): 357–365.

Hall, S. L., and Padgett, R. A. (1996). Requirement of U12 snRNA for in vivo splicing of a minor class of eukaryotic nuclear pre-mRNA introns. *Science* 271(5256): 1716–1718.

Henderson, J., Salzberg, S., and Fasman, K. (1997). Finding genes in DNA with a hidden Markov model. *J. Comput. Biol.* 4: 127–141.

Heinemeyer, T., Wingender, E., Reuter, I., Hermjakob, H., Kel, A., Kel, O., Ignatieva, E., Ananko, E., Podkolodnaya, O., Kolpakov, F., Podkolodny, N., and Kolchanov, N. (1998). Databases on transcriptional regulation: TRANSFAC, TRRD and COMPEL. *Nucl. Acids Res.* 26: 362–367.

Hodge, M. R., and Cumsky, M. G. (1989). Splicing of a yeast intron containing an unusual 5′ junction sequence. *Mol. Cell. Biol.* 9(6): 2765–2770.

Huang, X. O., Hardison, R. C., and Miller, W. (1990). A space-efficient algorithm for local similarities. *Comput. Appl. Biosci.* 6: 373–381.

Hutchinson, G. (1996). The prediction of vertebrate promoter regions using differential hexamer frequency analysis. *Comput. Appl. Biosci.* 12: 391–398.

Hutchinson, G. B., and Hayden, M. R. (1992). The prediction of exons through an analysis of splicible open reading frames. *Nucl. Acids Res.* 20: 3453–3462.

Iseli, C., Jongeneel, V., and Bucher, P. (1999). ESTScan: A program for detecting, evaluating, and reconstructing potential coding regions in EST sequences. *Ismb.* 138–148.

Jackson, I. J. (1991). A reappraisal of non-consensus mRNA splice sites. *Nucl. Acids Res.* 19(14): 3795–3798.

Jeffrey, H. J. (1990). Chaos game representation of gene structure. *Nucl. Acids Res.* 18: 2163–2170.

Kel, O., Romaschenko, A., Kel, A., Wingender, E., and Kolchanov, N. (1995). A compilation of composite regulatory elements affecting gene transcription in vertebrates. *Nucl. Acids Res.* 23: 4097–4103.

Knudsen, S. (1999). Promoter 2.0: For the recognition of PolII promoter sequences. *Bioinformatics* 15: 356–361.

Kohrman, D. C., Harris, J. B., and Meisler, M. H. (1996). Mutation detection in the med and medJ alleles of the sodium channel Scn8a. Unusual splicing due to a minor class AT-AC intron. *J. Biol. Chem.* 271(29): 17576–17581.

Kolossova, I., and Padgett, R. A. (1997). U11 snRNA interacts in vivo with the 5′ splice site of U12-dependent (AU-AC) pre-mRNA introns. *RNA* 3(3): 227–233.

Kondrakhin, Y. V., Shamin, V. V., and Kolchanov, N. A. (1994). Construction of a generalized consensus matrix for recognition of vertebrate pre-mRNA 3′-terminal processing sites. *Comput. Applic. Biosci.* 10: 597–603.

Krogh, A., Mian, I. S., and Haussler, D. (1994). A hidden Markov model that finds genes in E. coli DNA. *Nucl. Acids Res.* 22: 4768–4778.

Krogh, A. (1997). Two methods for improving performance of an HMM and their application for gene finding. *Ismb.* 5: 179–186.

Krogh, A. (2000). Using database matches with HMMgene for automated gene detection in Drosophila. *Genome Res.*, in press.

Kulp, D., Haussler, D., Rees, M., and Eeckman, F. (1996). A generalized hidden Markov model for the recognition of human genes in DNA. In *Proceedings of the Fourth International Conference on Intelligent Systems for Molecular Biology*, States, D., Agarwal, P., Gaasterland, T., Hunter, L. and Smith, R., eds. St. Louis: AAAI Press, 134–142.

Lapedes, A., Barnes, C., Burks, C., Farber, R., and Sirotkin, K. (1988). Application of neural network and other machine learning algorithms to DNA sequence analysis. *Proceedings Santa Fe Institute* 7: 157–182.

Lukashin, A. V., and Borodovsky, M. (1998). GeneMark.hmm: New solutions for gene finding. *Nucl. Acids Res.* 26: 1107–1115.

Manley, J. L. (1995). A complex protein assembly catalyzes polyadenylation of mRNA precursors. *Curr. Opin. Genet. Develop.* 5: 222–228.

Mathews, B. W. (1975). Comparison of the predicted and observed secondary structure of T4 phage lysozyme. *Biochem. Biophys. Acta.* 405: 442–451.

McLauchlan, J., Gaffney, D., Whitton, J. L., and Clements, J. B. (1985). The consensus sequence YGTGTTYY located downstream from the AATAAA signal is required for efficient formation of mRNA 3′ termini. *Nucl. Acids Res.* 13: 1347–1367.

Milanesi, L., and Rogozin, I. B. Prediction of human gene structure. In *Guide to Human Genome Computing*. 2nd ed., Bishop, M. J., ed. Cambridge: Academic Press, 215–259.

Mount, S. (1981). A catalogue of splice junction sequences. *Nucl. Acids Res.* 10: 459–472.

Mount, S. M. (1993). Messenger RNA splicing signal in Drosophila genes. In An Atlas of *Drosophila Genes*, Maroni, G., ed., 333–358. New York: Oxford University Press.

Mural, R. J., Mann, R. C., and Uberbacher, E. C. (1990). Pattern recognition in DNA sequences: The intron-exon junction problem. In *The first International Conference on Electrophoresis, Supercomputing and the Human Genome*. Cantor, C. R., and Lim, H. A., eds. London: World Scientific, 164–172.

Nakata, K., Kanehisa, M., and DeLisi, C. (1985). Prediction of splice junctions in mRNA sequences. *Nucl. Acids Res.* 13: 5327–5340.

Nilsen, T. W. (1994). RNA-RNA interactions in the spliceosome: Unraveling the ties that bind. *Cell* 78: 1–4.

Ohler, U., Harbeck, S., Niemann, H., Noth, E., and Reese, M. (1999). Interpolated Markov chains for eukaryotic promoter recognition. *Bioinformatics* 15: 362–369.

Parker, R., and Siliciano, P. G. (1993). Evidence for an essential non-Watson-Crick interaction between the first and last nucleotides of a nuclear pre-mRNA intron. *Nature* 361(6413): 660–662.

Pedersen, A. G., Baldi, P., Chauvin, Y., and Brunak, S. (1999). The biology of eukaryotic promoter prediction—a review. *Comput. Chem.* 23: 191–207.

Penotti, F. E. (1991). Human pre-mRNA splicing signals. *J. Theor. Biol.* 150: 385–420.

Perier, C. R., Praz, V., Junier, T., Bonnard, C., and Bucher, P. (2000). The eukaryotic promoter database (EPD). *Nucl. Acid Res.* 28: 302–303.

Prestridge, D., and Burks, C. (1993). The density of transcriptional elements in promoter and non-promoter sequences. *Hum. Mol. Genet.* 2: 1449–1453.

Proudfoot, N. J. (1991). Poly(A) signals. *Cell* 64: 617–674.

Rabiner, L. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proc. IEEE.* 77(2): 257–285.

Rabiner, L., and Juang, B. (1993). *Fundamentals of Speech Recognition.* New Jersey: Prentice Hall.

Reese, M. G., Harris, N. L., and Eeckman, F. H. (1996). Large scale sequencing specific neural networks for promoter and splice site recognition. In *Biocomputing: Proceedings of the 1996 Pacific Symposium*, Hunter, L., and Klein, T., eds. Singapore: World Scientific Publishing Co.

Reese, M. G., Hartzell, G., Harris, N. L., Ohler, U., and Lewis, S. E. (2000a). Genome annotation assessment in *Drosophila melanogaster. Genome Res., in press.*

Reese, M., Kulp, D., Tammana, H. L., and Haussler, D. (2000b). Genie—Gene finding in Drosophila melanogaster. *Genome Research, in press.*

Rogozin, I. B., and Milanesi, I. (1997). Analysis of donor splice sites in different eukaryotic organisms. *J. Mol. Evol.* 45: 50–59.

Salamov, A. A., and Solovyev, V. V. (1997). Recognition of 3′-end cleavage and polyadenilation region of human mRNA precursors. *CABIOS* 13(1): 23–28.

Salamov, A., and Solovyev, V. (1998). Fgenesh multiple gene prediction program: httpd://genomic.sanger.ac.uk.

Salamov, A., and Solovyev, V. (2000). Ab initio gene finding in Drosophila genomic DNA. *Genome Research* 10: 516–522.

Salsberg, S., Delcher, A., Fasman, K., and Henderson, J. (1998). A decision tree system for finding genes in DNA. *J. Comp. Biol.* 5: 667–680.

Seledtsov, I., and Solovyev, V. (1999). Genes in pictures: Interactive Java viewer for Infogene database. http://genomi.sanger.ac.uk/infodb.shtml.

Senapathy, P., Sahpiro, M., and Harris, N. (1990). Splice junctions, brunch point sites, and exons: Sequence statistics, identification, and application to genome project. *Methods in Enzymology* 183: 252–278.

Shahmuridov, K. A., Kolchanov, N. A., Solovyev, V. V., and Ratner, V. A. (1986). Enhancer-like structures in middle repetitive sequences of the eukaryotic genomes. *Genetics (Russ)* 22: 357–368.

Shapiro, M. B., and Senapathy, P. (1987). RNA splice junctions of different classes of eukaryotes: Sequence statistics and functional implications in gene expression. *Nucl. Acids Res.* 15(17): 7155–7174.

Sharp, P. A., and Burge, C. B. (1997). Classification of introns: U2-type or U12-type. *Cell* 91(7): 875–879.

Shepherd, J. C. W. (1981). Method to determine the reading frame of a protein from the purine/pyrimidine genome sequence and its possible evolutionary justification. *Proc. Natl. Acad. Sci. USA* 78: 1596–1600.

Snyder, E. E., and Stormo, G. D. (1993). Identification of coding regions in genomic DNA sequences: An application of dynamic programming and neural networks. *Nucl. Acids Res.* 21: 607–613.

Solovyev, V. V. (1993). Fractal graphical representation and analysis of DNA and Protein sequences. *BioSystems* 30: 137–160.

Solovyev, V. (1997). Fgenes multiple gene prediction program: httpd://genomic.sanger.ac.uk /gf.html.

Solovyev, V., and Kolchanov, N. (1994). Search for functional sites using consensus. In *Computer Analysis of Genetic Macromolecules: Structure, Function and Evolution.* Kolchanov, N. A., Lim H. A., eds. 16–21. Singapore: World Scientific Publishing Co.

Solovyev, V. V., Korolev, S. V., Tumanyan, V. G., and Lim, H. A. (1991). A new approach to classification of DNA regions based on fractal representation of functionally similar sequences. *Proc. Natl. Acad. Sci. USSR (Russ)(Biochem)*, 319(6): 1496–1500.

Solovyev, V. V., and Lawrence, C. B. (1993a). Identification of Human gene functional regions based on oligonucleotide composition. In *Proceedings of First International conference on Intelligent System for Molecular Biology*, Hunter, L., Searls, D., and Shalvic, J., eds. Bethesda, 371 379.

Solovyev, V., and Lawrence, C. (1993b). Prediction of human gene structure using dynamic programming and oligonucleotide composition In: *Abstracts of the 4th annual Keck symposium*. Pittsburgh, 47.

Solovyev, V., and Salamov, A. (1999a). INFOGENE: A database of known gene structures and predicted genes and proteins in sequences of genome sequencing projects. *Nucl. Acid Res.* 27: 248–250.

Solovyev, V., and Salamov, A. (1999b). HMM based prediction of CDS/ORF in mRNA/EST sequences: http://www.softberry.com/nucleo.html.

Solovyev, V. V., and Salamov, A. A. (1999). INFOGENE: A database of known gene structures and predicted genes and proteins in sequences of genome sequencing projects. *Nucl. Acids Res.* 27(1): 248 250.

Solovyev, V. V., Salamov, A. A., and Lawrence, C. B. (1994a). Predicting internal exons by oligonucleotide composition and discriminant analysis of spliceable open reading frames. *Nucl. Acids Res.* 22: 6156 5153.

Solovyev, V. V., Salamov, A. A., and Lawrence, C. B. (1994b). The prediction of human exons by oligonucleotide composition and discriminant analysis of spliceable open reading frames. In *Proceedings of the Second International Conference on Intelligent Systems for Molecular Biology*, Altman, R., Brutlag, D., Karp, P., Lathrop, R., and Searls, D., eds. Stanford, CA, 354–362.

Solovyev, V. V., Salamov, A. A., and Lawrence, C. B. (1995). Prediction of human gene structure using linear discriminant functions and dynamic programming. In *Proceedings of the Third International Conference on Intelligent Systems for Molecular Biology*, Rawling, C., Clark, D., Altman, R., Hunter, L., Lengauer, T., and Wodak, S., eds. Cambridge, Eng.: AAAI Press, 367–375.

Solovyev, V. V., and Salamov, A. A. (1997). The Gene-Finder computer tools for analysis of human and model organisms genome sequences. In *Proceedings of the Fifth International Conference on Intelligent Systems for Molecular Biology*, Rawling, C., Clark, D., Altman, R., Hunter, L., Lengauer, T., and Wodak, S., eds. Halkidiki, Greece: AAAI Press, 294–302.

Staden, R. (1984). Computer methods to locate signals in nucleic acid sequences. *Nucl. Acids Res.* 12: 505–519.

Staden, R. (1984). Mesurements of the effects that coding for a protein has on a DNA sequence and their use for finding genes. *Nucl. Acids Res.* 12: 551–567.

Staden, R., and McLachlan, A. (1982). Codon prefernce and its use in identifying protein coding regions in long DNA sequences. *Nucl. Acids Res.* 10: 141–156.

Stoesser, G., Tuli, M., Lopez, R., and Sterk, P. (1999). The EMBL nucleotide sequence dabase. *Nucl. Acids Res.* 27(1): 18–24.

Stormo, G. D., Schneider, T. D., Gold, L., and Ehrenfeucht, A. (1982). Use of the "Perceptron" algorithm to distinguish translational initiation sites in *E. coli*. *Nucl. Acids Res.* 10: 2997–3011.

Stormo, G. D., and Haussler, D. (1994). Optimally parsing a sequence into different classes based on multiple types of evidence. In *Proceedings of the Second International Conference on Intelligent Systems for Molecular Biology*, Altman, R., Brutlag, P., Karp, P., Lathrop, R., and Searls, D., eds. Menlo Park, CA: AAAI Press, 47–55.

Tabaska, J., and Zhang, M. Q. (1999). Detection of polyadenilation signals in Human DNA sequences. *Gene* 231: 77–86.

Tarn, W. Y., and Steitz, J. A. (1996a). A novel spliceosome containing U11, U12, and U5 snRNPs excises a minor class (AT-AC) intron in vitro. *Cell* 84(5): 801–811.

Tarn, W. Y., and Steitz, J. A. (1996b). Highly diverged U4 and U6 small nuclear RNAs required for splicing rare AT-AC introns. *Science* 273(5283): 1824–1832.

Tarn, W. Y., and Steitz, J. A. (1997). Pre-mRNA splicing: The discovery of a new spliceosome doubles the challenge. *Trends Biochem. Sci.* 22(4): 132–137.

Thanaraj, T. A. (1999). A clean data set of EST-confirmed splice sites from Homo sapiens and canonicals for clean-up procedures. *Nucl. Acids Res.* 27(13): 2627–2637.

Thanaraj, T. A. (2000). Positional characterization of false positives from computational prediction of human splice sites. *Nucl. Acids Res.* 28: 744–754.

Thomas, A., and Skolnick, M. (1994). A probabilistic model for detecting coding regions in DNA sequences. *IMA J. Math. Appl. Med. Biol.* 11: 149–160.

Tjian, R., and Maniatis, T. (1994). Transcriptional activation: A complex puzzle with few easy pieces. *Cell* 77: 5–8.

Uberbacher, E., and Mural, J. (1991). Locating protein coding regions in human DNA sequences by a multiple sensorneural network approach. *Proc. Natl. Acad. Sci. USA* 88: 11261–11265.

Wadman, M. (1998). Rough draft of human genome wins researchers' backing. *Nature* 393: 399–400.

Wahle, E. (1995). 3′-end cleavage and polyadelanytion of mRNA precursor. *Biochim. Biophys. Acta* 1261: 183–194.

Wahle, E., and Keller, W. (1992). The biochemistry of the 3′-end cleavage and polyadenylation of mRNA precursors. *Annu. Rev. Biochem.* 61: 419–440.

Werner, T. (1999). Models for prediction and recognition of eukaryotic promoters. *Mammalian Genome* 10: 168–175.

Wieringa, B., Hofer, E., and Weissmann, C. (1984). A minimal intron length but no specific internal sequence is required for splicing the large rabbit Bglobin intron. *Cell* 37: 915–925.

Wieringa, B., Meyer, F., Reiser, J., and Weissmann, C. (1983). Unusual splice sites revealed by mutagenic inactivation of an authentic splice site of the rabbit beta-globin gene. *Nature* 301(5895): 38–43.

Wilusz, J., Shenk, T., Takagaki, Y., and Manley, J. L. (1990). A multicomponent complex is required for the AAUAAA-dependent cross-linking of a 64-kilodalton protein to polyadenylation substrates. *Mol. Cell. Biol.* 10: 1244–1248.

Wingender, E., Dietze, P., Karas, H., and Knuppel, R. (1996). TRANSFAC: A database of transcription factors and their binding sites. *Nucl. Acid. Res.* 24: 238–241.

Wingender, E. (1988). Compilation of transcription regulating proteins. *Nucl. Acids Res.* 16: 1879–1902.

Wu, Q., and Krainer, A. R. (1997). Splicing of a divergent subclass of AT-AC introns requires the major spliceosomal snRNAs. *RNA* 3(6): 586–601.

Xu, Y., Einstein, J. R., Mural, R. J., Shah, M., and Uberbacher, E. C. (1994). An improved system for exon recognition and gene modeling in human DNA sequences. In *Proceedings of the 2nd International Conference on Intelligent Systems for Molecular Biology*, Altman, R., Brutlag, Karp, P., Lathrop, R., and Searls, D., eds., Cambridge, UK: AAAI Press, 376–383.

Xu, Y., and Uberbacher, E. C. (1996). Gene prediction by pattern recognition and homology serach. In *Proceedings of the 4th International Conference on Intelligent Systems for Molecular Biology*, 4: 241–251.

Xu, Y., Mural, R. J., and Uberbacher, E. C. (1997). Inferring gene structures in genomic sequences using pattern recognition and expressed sequence tags. *ISMB* 5: 344–353.

Yada, T., Ishikawa, M., Totoki, Y., and Okubo, K. (1994). Statistical analysis of human DNA sequences in the vicinity of poly(A) signal. Institute for New Generation Computer Technology, technical report TR-876.

Yu, Y. T., and Steitz, J. A. (1997). Site-specific crosslinking of mammalian U11 and u6atac to the 5′ splice site of an AT-AC intron. *Proc. Natl. Acad. Sci., USA* 94(12): 6030–6035.

Zhang, M. Q., and Marr, T. G. (1993). A weight array method for splicing signal analysis. *Comp. Appl. Biol. Sci.* 9(5): 499–509.

Zhang, M. Q. (1997). Identification of protein coding regions in the human genome by quadratic discriminant analysis. *Proc. Natl. Acad. Sci. USA* 94: 565–568.

10 Computational Methods for Promoter Recognition

Michael Q. Zhang

10.1 Introduction

In this chapter, we shall describe the problem of promoter recognition. We begin with a brief introduction to the biology of promoter structure and function. We then review some of the current computational approaches to the problem, with emphasis on basic concepts and methodologies in real applications. Interested readers should consult the references for more technical details or program specifications.

There are two main classes of functional information encoded in the genomic DNA of every living organism. One class is the coding regions, which specify the structure and function of each gene product; the other class is the regulatory regions (occasionally, but very rarely, overlapping with a coding region), which control and regulate when, where, and how the genes are expressed. Promoter is the most important regulatory region that controls and regulates the very first step of gene expression: mRNA transcription. For a comprehensive review on the related biology, see the excellent book *Transcriptional Regulation in Eukaryotes* by Carey and Smale (1999).

Promoter is commonly referred to as the DNA region that is required to control and regulate the transcriptional initiation of the immediately downstream gene. For a typical eukaryotic (PolII or protein-coding) gene, it contains a core promoter of about 100 bp centered around the transcriptional start site (TSS), and a proximal promoter of about 500 bp immediately upstream of the core promoter. Often complex regulation in vivo can involve many more features, such as enhancers, locus control regions (LCRs), and/or scaffold/matrix attachment regions (S/MARs). Some people refer to enhancers as the distal promoter elements, which can be either upstream or downstream of the gene or within an intron and can be in any orientation. For our purpose, we use the region (−500, +100) with respect to a TSS as a specific definition.

The main characteristic of a promoter is that it contains aggregates of transcription factor (TF) binding sites. During the process of development, genes are turned on and off in a preprogrammed fashion, a process that eventually generates cell specificity. This developmental program is orchestrated by TFs, which bind to specific DNA sites in the promoters near genes they control. A single TF is not dedicated to each regulatory event. Instead, different combinations of ubiquitous and cell-specific regulatory factors are used to achieve a combinatorial control.

Core promoter, approximately in (−50, +50), (1) binds to and controls assembly of the preinitiation complex (PIC) containing PolII, the general transcription factor (GTF), and coactivators; (2) positions the TSS and controls the direction of transcription; and (3) responds to nearby or distal activators (we use the same terms,

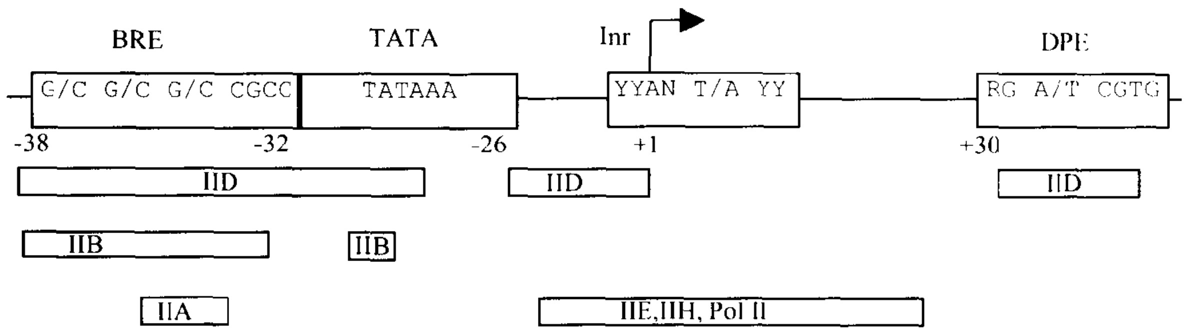


Figure 10.1
Sequence elements and GTF footprints in a typical core promoter.

"activators" or "enhancers," to also imply "repressors" and "silencers," depending on the context, for simplicity) binding proximal promoter and enhancers. The PIC comprises the GFTs (PolII, TFIIA, TFIIB, TFIID, TFIIE, TFIIF, and TFIIH) and coactivators that mediate response to regulatory signals. A typical core promoter contains four DNA elements (figure 10.1): TATA-box (binding site for the TBP subunit of TFIID. Although recently some alternative TATA binding proteins have been discovered in a few specific types of cells (see Holmes and Tjian 2000), this review deals with only the major types of TBP: Inr (overlapping with TSS), DPE (downstream core promoter element), and BRE (TFIIB recognition element). Not every element occurs in a core promoter. People have classified core promoters according to the presence or the absence of TATA and/or Inr elements (reviewed in Novina and Roy 1996). Many "housekeeping gene" core promoters appear to lack both TATA and Inr elements but instead contain several TSSs, a high G+C content, and multiple binding sites for the ubiquitous TF Sp1 (Smale 1994), which directs the formation of PIC to a region 40–100 bp downstream of its binding sites. Purified GTFs and PolII mediate basal (low-level) transcription on a core promoter in vitro but cannot support activated transcription in the absence of coactivators. More recent studies indicate that the functional form of PIC in vivo must also include coactivators/mediators. The interaction of activators with any surface of this large GTF-containing complex (also called holoenzyme, reviewed in Parvin and Young 1998) allows recruitment of the complex to the core promoter and response of the polymerase to the regulatory signals.

Transcriptional regulation is controlled by the binding of sequence-specific DNA-binding TFs to proximal promoters, approximately in (−500, −50) (also called regulatory promoters), and enhancers (reviewed in Blackwood and Kadonaga 1998). It should be noticed that there is no real distinction between proximal and distal

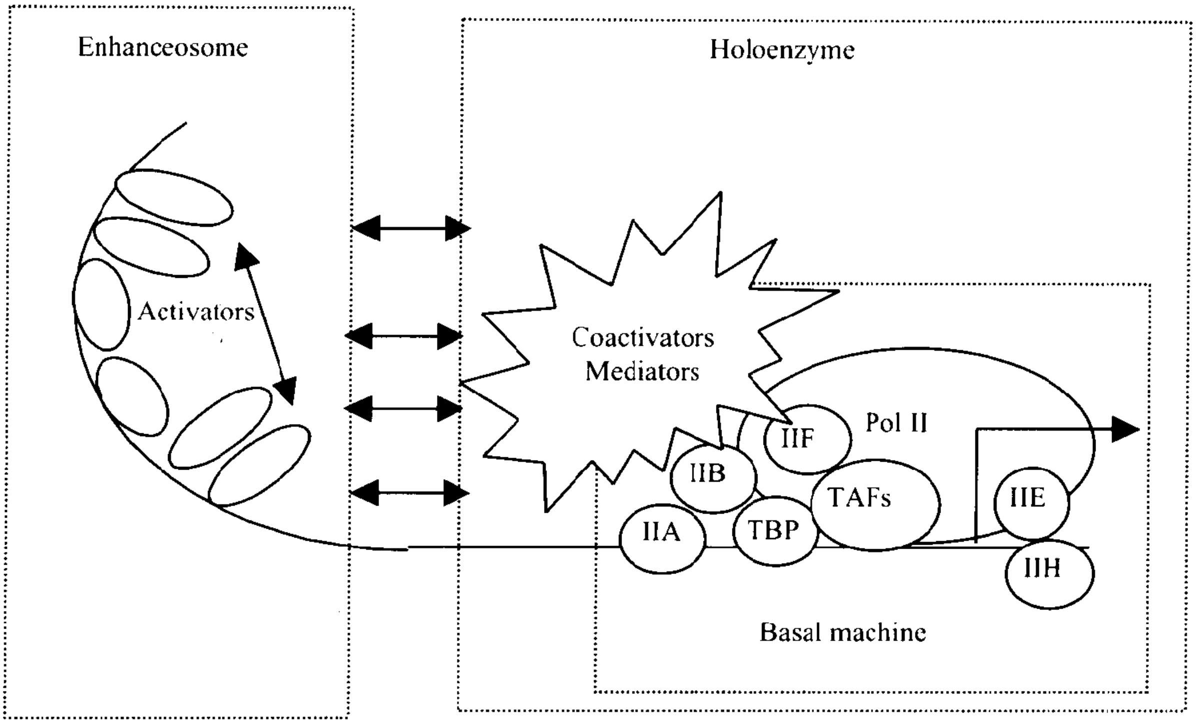


Figure 10.2
Recruitment and activation of the PIC.

(enhancer) regulatory elements; they often involve the same set of TF binding sites. Some cooperative binding of activators to enhancers and proximal promoters can lead to the assembly of nucleoprotein structures termed "enhanceosomes" (figure 10.2; see also Thanos and Maniatis 1995).

In a living eukaryotic cell, DNA is not naked; instead, it is wrapped into nucleosomes by histones. With the help of many other non-histone proteins (NHP), nucleosomes are further condensed into chromatin filament. These higher order structures are believed to be necessary to keep most genes in a (default) repressed state. To activate a gene, the chromatin encompassing that gene and its control regions must be altered or "remodeled" to permit TFs to access their specific binding sites. Because of the complexity of such long-range interaction among many global regulators, chromatin remodeling is beyond the reach of current promoter recognition algorithms. Therefore, all existing computational methods implicitly assume all TF sites are accessible, which is the intrinsic source of a large number of false positives.

In summary, promoter is the key DNA region that controls and regulates transcription. Delineation of the promoter architecture is fundamental for understanding gene expression patterns, regulation networks, cell specificity, and development. It is

also important for designing efficient expression vectors or to target specific delivery systems in gene therapy. In the large-scale genomic sequencing era, promoter prediction is also crucial for gene discovery and annotation.

There have been many computational approaches to this extremely difficult problem. I recommend some recent reviews (e.g., Pedersen et al. 1999; Werner 1999; Fickett and Wasserman 2000; Stormo 2000) where one can find further references.

Depending on the goals, computational approaches can be divided into two classes: general promoter recognition methods and specific promoter recognition methods. The primary goal for the general methods is to identify TSS and/or core promoter elements for all genes in a genome; the specific methods focus on identifying specific regulatory elements (TF sites) that are shared by a particular set of transcriptionally related genes. Specific methods can have very high specificity when searching against the whole genome and can provide immediate functional clues to the downstream gene. But because of their broad coverage, the general methods are extremely useful for large-scale genome annotation. I shall first describe the specific promoter recognition problem, which is how to find functional TF sites. I shall then take on the general problem, how to discriminate a promoter region from other genomic regions.

10.2 Finding Transcription Factor (TF) Binding Sites

10.2.1 Site, Consensus, and Weight Matrix

As a specific promoter class is characterized by a specific set of TFs, finding TF binding sites is the most important step in promoter recognition. There are at least two classes of TFs (from now on, we use TF to refer to DNA binding transcription factors). One class is the general or ubiquitous TFs, such as TATA-box binding protein (TBP) or Sp1. Their binding sites can be identified by simply collecting a large number of promoter (−500, +50) sequences. The other class is the specific TFs, which can only be identified by getting a specific set of promoters that share the same site (i.e., their target genes are co-regulated by the same TF). Experimentally, biologists are able to identify a TF site *de novo* with a single promoter sequence. They can characterize such a site by mutagensis and obtain a *consensus* description (such as the *E. coli* TATA-box TATAAT or allowing degeneracy TATRNT). As more sites are known, one can get the same information by aligning the sites. Although it is easy to write a consensus pattern to represent aligned sites, it is difficult to find one that is optimal for predicting the occurrence of new sites (generalizability) or for discriminantly ranking the binding activities of the sites (differentiability). In most applications, a position *weight matrix* (PWM) is often superior.

10.2.2 Constructing a Matrix Given the Alignment

To explain the high degeneracy found in many sites, Berg and von Hippel (1987) proposed a theory on the selection of DNA binding sites by regulatory proteins. They assumed that specific sites have been selected according to some functional constraint (e.g., the binding affinity or activity must be in some tolerable range), and all sequences that can fulfill this requirement are equally likely to occur. This theory, which has been checked by experiments, provided a link between a natural scoring function (minus binding energy) and observed base frequency at each position:

$$S_{bx} = \log(p_{bx}/p_{0x}) \tag{10.1}$$

where s_{bx} is the score (PWM element) for a base "b" at position "x," p_{bx} is the frequency of the base "b" found at "x" in the site, and "0" indicates the base (consensus base) that has the lowest energy. Often it is desirable to use a scoring function that can best discriminate the set of target sites from a set of control sites. In this situation, we replace p_{0x} by p_b^0, the base frequency evaluated in the control set (here, we assume they do not depend on the position). Choosing p_b^0 appropriately (representing the correct background contrast) can be very important for searching other such sites in a genome. The total score of the site is the sum of individual position scores. Using a single base frequency implies the assumption of independence between any pair of bases, although this can be easily generalized to high-order Markov models (e.g., Zhang and Marr 1993).

It should be mentioned that to construct a statistical measure, the quality of sequences is obviously extremely crucial. In addition to the integrity of the data, statistical independence of the sequences is essential so that the result will not be biased by the sample. There are many ways to reduce the redundancy; a simple criterion may be that no pair of sequences should have more than 90 percent identity within the 100-bp surrounding region.

There are many ways to introduce pseudo-counts in order to avoid a null frequency. This is equivalent to introducing a prior probability, which is necessary when an observed count is rare. The Laplace plus-one method (i.e., add one to every base count at every position, and hence correspond to a uniform prior) is a simple and popular choice.

How to determine the length of a motif can be a very subtle problem. Conventionally, one uses the relative information to measure the significance of each position (Schneider et al. 1986):

$$I_x = \Sigma_b p_{bx} \log(p_{bx}/p_b^0) = \Sigma_b p_{bx} s_{bx} \tag{10.2}$$

which obviously has the meaning of the average binding energy of all the known sites at position "x." The optimal length of the motif can be obtained by an optimization procedure. One procedure, consistent with the Berg and von Hippel theory, would be to find an optimal window site so that the total information within the window (the area under the curve of I_x) minus the average of the total information in the two flanking windows become maximum. It can be further refined by the discriminant procedure described below.

10.2.3 Searching for a Known Binding Site

Given a motif, either in the form of a consensus or a matrix, one must first assess the quality of the motif and determine a threshold value before one can use it to search for new members of the site. The way to do this is to perform a standard classification test (e.g., Fukunaga 1990) in which both the threshold score and the motif length may be optimized by minimizing the classification (Bayesian) error. Because a single TF site does not have enough specific information due to its short length (about 5 to 25 bp) and high degeneracy, any unconstrained genome search will almost certainly result in a lot of false positives. The specificity can only be achieved by combining interactive (correlated) sites into a promoter module (also called a composite site) and by higher order structure constraints (long-range control elements).

10.3 Identifying Motifs with Unaligned Sequences

In order to discover novel motif sites, one has to use more sophisticated approaches. Given a set of related sequences, these methods must be able to find motif(s) shared by majority of the sequences and statistically significant, in a reasonable time. Although we are focusing on TF sites, all methods should be applicable to more general sequence motif discovery problems. There are existing numerous algorithms (see reviews: e.g., Vanet et al. 1999; Brazma et al. 1998; Pesole et al. 1996; French et al. 1997). Below is necessarily a personal selection that represents a short list of generic methodologies. HMM (Hidden Markov Model see e.g., Durbin et al. 1998) and neural network (see e.g., Baldi and Brunak 1998) are not included because there are special books describing these machine learning approaches.

10.3.1 *k*-tuple or Exhaustive Pattern Search Methods

For finding short and highly conserved motifs (such as many typical TF sites in yeast), *k*-tuple-based methods can be very effective. The basic idea is to detect over-represented (with respect to a control set or to a background set) *k*-tuples systematically.

Relative Information (RI) The simplest method is to calculate the frequencies for all k-tuples in both the target set and the control set. k is usually limited by the size of the sample data ($4^k \leq (L - k + 1)^* N$, where L is the length of each sequence and N is the number of sequences). One can define a RI for every word w of length k as (also called LOG-ODD Ratio, or negative relative entropy):

$$\mathrm{RI}(w) = \log(f(w)/f^0(w)) \tag{10.3}$$

Although the exact statistic is not known, one can use $z = (\mathrm{RI} - \mathrm{mean}(\mathrm{RI}))/\mathrm{std}(\mathrm{RI})$ to estimate the significance. If f^0 represents a random background (usually a Markov approximation of order less than k), one could use chi-square test on $\chi^2 = \Sigma_w[O(w) - E(w)]^2/E(w)$, where $O(w)$ is the observed number of w and $E(w)$ is the expected number (calculated using f^0). This has been applied in yeast promoter analysis (Zhang 1999, and see 10.3.5 below). This method can be easily generalized to allow limited degeneracy and/or iterative extension of the k-tuple motif (see Zhu and Zhang 2000, where a motif pattern was defined as a 6-tuple, allowing up to one mismatch and an iterative procedure for extending such a motif using the χ^2-test).

WORDUP This is a similar method but requiring the motif to be shared by majority of the sequences (Pesole et al. 1992). The statistical significance of each k-tuple word w is determined by comparing, through a χ^2-test, the actual number of different sequences in which w is present with the expected occurrences. Expectations are calculated on the basis of two assumptions: (1) oligonucleotides are Poisson distributed, and (2) nucleotide sequences can be generated according to a first order Markov chain. Because the probability $p_i(w)$ that w is found at least once in the ith sequence is $p_i(w) = 1 - \exp[-\lambda_i(w)]$ with $\lambda_i(w) = p_i^0(w)(L_i - k + 1)$ and $p_i^0(w)$ is the Markov approximation of $f_i(w)$, namely $p_i^0(w) = f(w_{1,2})f(w_{2,3})\dots f(w_{k-1,k})/f(w_2)f(w_3)\dots f(w_{k-1})$. The expected number of sequences containing w is given by $E(w) = \Sigma_i p_i(w)$. If $O(w)$ is the observed number of sequences containing w, the standard χ^2-value given above can be used to rank significant k-tuples (the default threshold is 20) that form a vocabulary. An iterative procedure was also used to construct a new vocabulary containing all significant words of length greater or equal to k (http://bigarea.area.ba.cnr.it:8000/EmbIT/coda_word.html).

A similar method was developed by van Helden et al. (1998) using slightly different statistical criterion. More sophisticated algorithms for detecting more complex patterns (with multiple sites) have also been developed recently (Marsan and Sagot 2000).

10.3.2 Multiple Sequence Alignment Methods

For longer and more degenerate motifs, one has to use multiple sequence (local) alignment algorithms. Given N (number of sequences), L (length of each sequence)

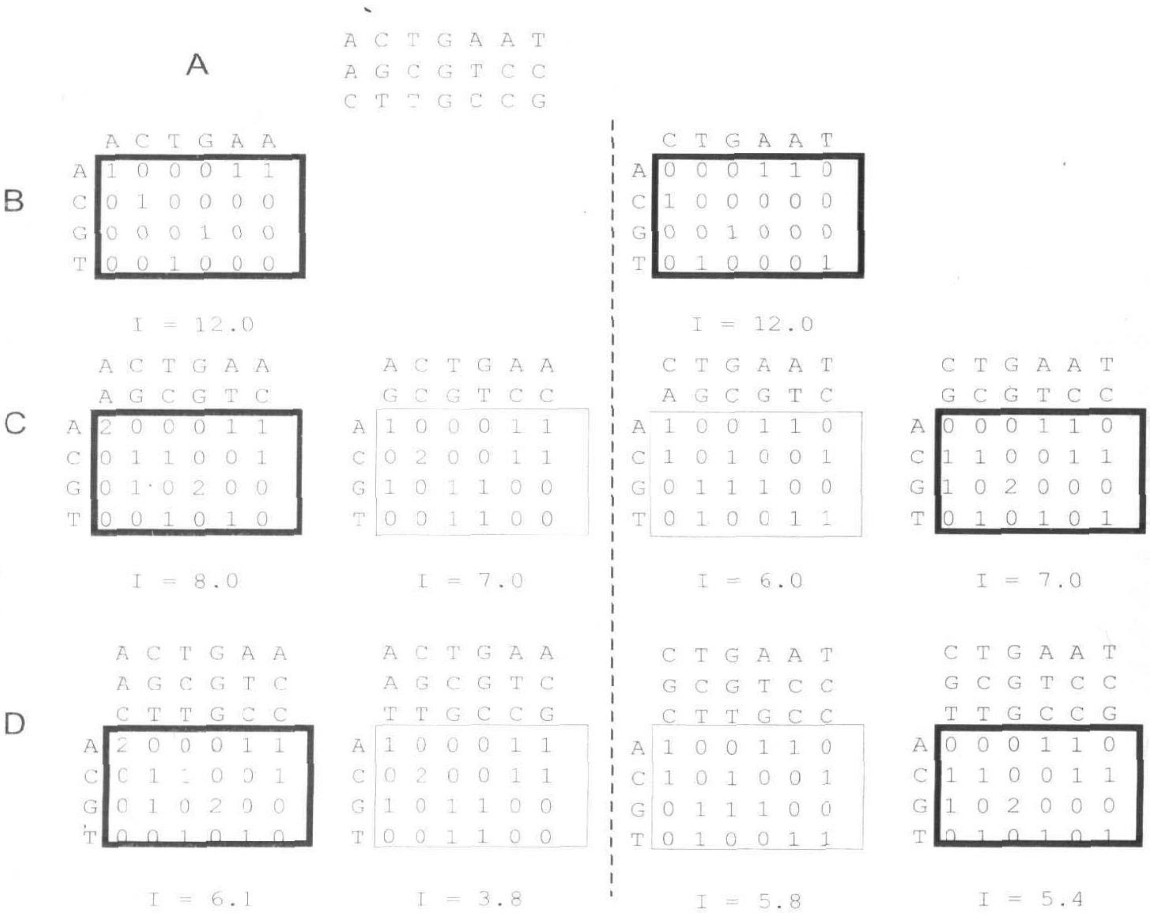


Figure 10.3
A diagram of the algorithm in CONSENSUS (Hertz et al. 1990).

and k (length of the motif with indels), there are $(L - k + 1)^N$ possible alignments. Finding an optimal alignment that maximizes an objective function (say, $I = \Sigma_x I_x$) is a hard problem. Various heuristic approaches are available to attack this multi-dimensional optimization problem. I describe here three generic methods: CONSENSUS (a greedy algorithm), EM/MEME (EM algorithms), and Gibbs sampler (a stochastic sampling algorithm).

CONSENSUS A greedy algorithm originally developed by Stormo and Hartzell (1989) and implemented in *CONSENSUS* (Hertz et al. 1990) this is a heuristic method, which is quite efficient and has been widely used in DNA motif discoveries.

The basic idea is illustrated in a toy example shown in figure 10.3. Given the three sequences (a) to be aligned, the algorithm starts by forming a frequency matrix for each of the k-tuples in the first sequence (b). Each of these matrices is then combined

with each k-tuple in the second sequence to form new matrices containing two k-tuples (c). However, for each k-tuple from the first sequence, the program only saves the "best" progeny matrix (measured by the information content). In the next cycle, each saved matrix is combined with each k-tuple in the third sequence to form new matrices, each containing three k-tuples (d). Again, the program only saves the best progeny of each matrix from the previous cycle. This cycle is repeated until the last sequence in the set has contributed a k-tuple to the saved matrices. Of the matrices saved after the last cycle, the one with the lowest probability of occurring by chance is considered to describe the consensus motif (the first in D). In practice, ties occur during the cycles, so that the number of matrices at the end is greater than the number of k-tuples in the first sequence.

The *CONSENSUS* program was first used to accurately identify the known consensus pattern for the *E. coli* CRP protein binding sites (Stormo and Hartzell 1989). It was then further improved and tested for robustness on the *E. coli* LexA protein binding sites (Hertz et al. 1990). In both cases, the order in which the sequences are presented is not critical (the latest version of the program allows the user to set a parameter so that the result will not depend on the input order at all). The program is also robust enough to tolerate some sequences that do not contain binding sites Thanks to the effort of Hertz, *CONSENSUS* has been constantly improved upon. Some of the important additions to the original algorithm are (1) independence of the input sequence order, (2) autodetecting motif length, (3) allowing limited insertions/deletions, and (4) more rigorous statistical evaluation of the p-value (Hertz and Stormo 1999).

EM and MEME EM (expectation maximization) is a standard technique widely used in maximum-likelihood estimations (Dempster et al. 1977). Expectation maximization algorithms are named for their two iterative steps, the expectation (E) step and the maximization (M) step, which are alternately repeated until a convergence criterion is satisfied. Lawrence and Reilly (1990) first developed an EM algorithm and tested it on cyclic adenosine monophosphate receptor protein (CRP) binding sites. One starts with an initial guess on the base probability p_{bx} within the sites and the background p_b^0 for the nonsites, then the probability of the event B_{jy} that the site begins at position y in sequence j can be calculated by Bayes formula:

$$P(B_{jy} \mid \mathbf{p}, S) = P(S \mid B_{jy}, \mathbf{p}) / \Sigma_x P(S \mid B_{jx}, \mathbf{p}) \tag{10.4}$$

where S is the sequence data and the prior probability $P^0(B_{jy})$ is uniform, that is, $1/(L-k+1)$ and $P(S \mid B_{jy}, \mathbf{p})$ = the product of probabilities for all bases. Using

formula 10.4, one can complete the E-step by calculating the expected number n_{bx} of the base b at position x in the site and the expected number n_b^0 of the base b in the nonsites. The M-step is simply to replace $p = \{p_{bx}, p_b^0\}$ by $\{n_{bx}/N, n_b^0/N\}$ (the maximum likelihood estimators). One then iterates this to convergence when the parameter estimates no longer change.

MEME (Bailey and Elkan 1994) added several extensions to EM to overcome some limitations. MEME chooses starting points systematically, based on all subsequences of the data. It eliminates the assumption of one motif per sequence and allows each sequence to contain zero, one, or several appearances of the shared motif. Furthermore, MEME probabilistically "erases" the appearances of a site after it is found, and continues searching for other shared motifs in the dataset. The newer version has made MEME smarter and more robust as an unsupervised motif-discovering tool, as it will automatically determine the motif length and/or choose whether or not to enforce the palindrome constraint (Bailey and Elkan 1995). Once a MEME motif is found, MAST (Bailey and Gribskov 1998) can be used to search other sequences for new members. Both MEME and MAST are available at http://www.sdsc.edu/MEME.

Gibbs Sampler As greedy or EM-based algorithms cannot guarantee to find the global maximum and may be prone to local optima, stochastic algorithms have been developed to overcome this problem. The Gibbs sampler, which consists of a site sampler (Lawrence et al. 1993) and a motif sampler (Neuwald et al. 1995), has been a very successful one.

The site sampler assumes every sequence contains at least one site. The algorithm is initialized by choosing random starting positions within all the sequences. It then proceeds through two steps of Gibbs sampler iteratively. First, it builds a model step by constructing a model $\mathbf{p} = \{p_{b,x}, p_b^0\}$, as in the EM case, using all the sequences with the selected sites except the first sequence. Then it samples a new site for the first sequence from every possible position according to a relative weight $p_{b,x}/p_b^0$. Then it repeats both steps for subsequent sequences. A cycle is complete when the site for the last sequence is re-sampled. Theoretically, after an infinite number of cycles, the relative information $I = \Sigma_x I_x$ (formula 10.2) will reach its maximum. In practice, the alignment often converges fairly fast. Sometimes the sampler can get stuck at suboptima, which may require a simultaneous shift of all the aligned sites to the left or to the right by few bases. To speed up the convergence, the Gibbs sampler automatically samples a shift again according to the relative weight of the likelihood ratio after a certain specified number of cycles. This basic algorithm was also generalized to allow more than one type of motif per sequence.

In order to find sites that may have multiple copies in some sequences and zero copies in others, the motif (or Bernoulli) sampler was designed to concatenate all the sequences into a single one. It is initialized to an alignment of sites randomly spread throughout (no site can overlap another or across a sequence boundary); the rest are nonsites. The algorithm starts with picking the first possible site position out of either the aligned set or the nonsite set, updates the model $p = \{p_{bx}, p_b^0\}$ (i.e., recalculates the base counts), then samples this position into the aligned set or the nonsite set according to the odd ratio $[\mathrm{P(site)}^* p_{bx}]/[\mathrm{P(nonsite)}^* p_b^0]$, where the posterior (prior + pseudo-counts) P(site) and P(nonsite) may be specified by a user. Then the algorithm continues with picking the second site, and so on. One cycle is complete when the last possible site position is sampled. Iterate enough cycles untill convergence of the motif alignment (or the maximum number of cycles specified). This has been generalized to handle more than one type of motif. Two other useful technical features are the column sampling (to allow automatically increasing the site length by sampling in more conserved flanking columns) and the near optimal sampling (to allow estimating relative probability, each site is sampled into the alignment). A Gibbs sampler server is currently maintained at http://bayesweb.wadsworth.org/gibbs/. Two other modified versions of the Gibbs sampler for DNA sequence analysis have been reported in microarray data analysis applications (AlignACE: Roth et al. 1998; and GibbsDNA: Zhang 1999a).

10.3.3 Statistical Significance

It is also important to know how significant a particular alignment is with respect to a random model. Because all the alignments are ranked by the relative information I, it would be desirable to calculate the p-value, namely, the probability of finding an alignment with relative information greater than or equal to I. Assuming the null model for each alignment column is an independent multinomial model:

$$P_{matrix} = \sum_{x=1}^{k} \frac{N!}{\prod_{b=A}^{T} n_{bx}} \prod_{b=A}^{T} p_b^{n_{bx}} \tag{10.5}$$

If I is small and N is large, $2NI$ tends to a χ^2 distribution (df = 3k). Unfortunately, promoter analyses generally involve very large scores and frequently few sequences, and the limiting distribution tends to give poor probability estimates. Using large-deviation technique, Hertz and Stormo (1999) obtained the approximate mathematical formula for the p-value and the E-value (expected number of alignments with I or greater). They also implemented an efficient algorithm for calculating these values in

CONSENSUS. Other methods have also been reported for estimating the statistical significance of a matrix search result (e.g., Staden 1989; Claverie and Audic 1996).

10.3.4 Constructing Regulatory Modules

Because promoter is regulated by TF modules made of composite sites, simultaneous detecting of correlated sites is much more significant and hence provides better specificity. Claverie and Sauvaget (1985) published one of the earliest methods for detecting two sites in a fixed distance and orientation in the heat-shock promoters.

Another interesting example was given by the identification of regulatory modules that confer muscle-specific gene expression (Wasserman and Fickett 1998), where a logistic regression analysis (LRA: Hosmer and Lemeshow 1989) was used to combine matrix scores for multiple TF sites in each module. This directly generalized the study of the two-site module (MEF2/MyoD model: Fickett 1996).

More recently, experimental analysis and computer prediction of CTF/NFI TF sites were reported (Roulet et al. 2000) where a generalized profile model for CTF-1 DNA binding specificity was proposed. This model consists of a conserved half-site (5 bp) + a spacer (5,6,7, or ∞) + a less conserved (palindromic) repeat. Detailed experimental analysis reveals the flexible and correlated nature of this protein binding site.

With detailed modeling of TF modules, one will be able to recognize promoters of a specific class with extremely high specificity (French et al. 1998). Unfortunately, generation of these models requires high quality as well as systematic experimental data, which are still very rare. The development of composite site databases such as COMPEL and TRRD (Heinemayer et al. 1998) will greatly facilitate advances in this field.

10.3.5 Large-Scale Gene Expression

Recent advent of large-scale gene expression technologies is having a great impact on the understanding of gene regulation (e.g., Schena et al. 1995; Lockhart et al. 1996). By clustering gene expression profiles, different groups of co-regulated genes can be identified and their promoter elements may be detected by either *k*-tuple (Zhu and Zhang 2000; van Helden et al. 1998) or multiple alignment (Zhang 1999a; Hughes et al. 2000) methods once the upstream sequences become available for the transcriptionally co-regulated genes (e.g., DeRisi et al. 1997; Spellman et al. 1998; Cho et al. 1998; Roth et al. 1998). Using large-scale expression data to detect novel promoters and to infer regulation networks will become the cutting-edge bioinformatics in the functional genomics era (Zhang 1999b; Bucher 1999; McGuire et al. 2000). Species-specific promoter databases such as SCPD (Zhu and Zhang 1999) shall become extremely useful resources for studying large-scale transcription data.

10.3.6 Phylogenetic Footprinting

As more genomes become available, comparative analysis of noncoding regions has also become an important approach for detecting promoters or regulatory regions in general (Aparicio et al. 1995; Gumucio et al. 1996; Jareborg et al. 1999). Phylogenetic footprinting is referred to (Fickett and Wasserman 2000) as the identification of any functional region by comparison of orthogous genomic sequences between species. Although the orthologous coding regions (ORFs) are highly conserved, the conservation of regulatory regions varies widely with particular genes. To detect short TF sites, one would want to compare orthologous regulatory regions between species that are not too close (so that the sequences have enough time to diverge) and not too distant (so that some related regulatory regions are still recognizable; see Duret and Bucher 1997). But we are currently limited to the few sequenced model systems. Several methods for detecting conserved blocks from a multple alignment have been evaluated by Stojanovich et al. (1999). Programs designed for very long alignments of syntenic regions have also become available (e.g., PIPmaker: Schwartz et al. 2000; MUMmer: Delcher et al. 1999). Among many applications, PIPmaker was very successfully used for the identification of a coordinate regulator of interleukins 4, 13, and 5 (Loots et al. 2000).

10.4 Predicting Transcriptional Start Site (TSS)

Because TF sites can occur anywhere, even with the location of a regulatory module in proximal promoter, identification of TSS is still not easy. General promoter prediction methods mainly focus on TSS site prediction in order to locate the beginning of a gene instead of seeking specific regulatory elements.

10.4.1 CpG Islands

Vertebrate genomic DNA is known to be generally depleted in the dinucleotide CpG. In the human genome, for example, the occurrence of CpG dinucleotides is five times less than statistically predicted from the nucleotide composition (Bird 1980). CpG depletion is believed to result from methylation of Cs at 80 percent CpG dinucleotides, which leads to mutation of the methylated C to T, and thus conversion of the CpG dinucleotides to TpG (Bird 1999). There are, however, genomic regions of high G+C content, termed CpG islands, where the level of methylation is significantly lower than the overall genome. In these regions, the occurrence of CpGs is significantly higher, close to the expected frequency. As defined by Gardiner-Garden and Frommer

(1987), CpG islands are greater than 200 bp in length, have more than 50 percent of G+C content, and have a ratio of CpG frequency to the product of the C and G frequencies above 0.6. The CpG island is an important signature of the 5′ region of many mammalian genes, often overlapping with, or within, a thousand bases downstream of the promoter (Cross and Bird 1995), where a promoter associate nucleosome is found (Ioshikhes et al. 1999). The identification of promoters by CpG islands with a resolution of 2 KB will be most useful for large-scale sequence annotation. Although visual inspection of CpG islands is often used for gene identification by many molecular biologists, Ioshikhes and Zhang (2000) recently optimized the features that can best discriminate the promoter-associate CpG islands from the non-associated ones. This led to an effective algorithm (CpG_Promoter) for large-scale promoter mapping with 2 KB resolution. Statistical tests showed that about 85 percent of CpG islands within an interval from −500 to +1500 around the TSS (transcriptional start site) were correctly identified, and that roughly 93 percent of the CpG island containing promoters were correctly mapped. The basic procedure is to use CpGPlot program of R. S. Lopez (available at http://www.sanger.ac.uk/Software/EMBOSS/, see Larsen et al. 1992) for mapping of potential CpG islands and then to use a Quadratic Discriminant classifier for prediction of promoter associated islands. The EMBL CPGISLE database of human CpG islands was used for training the classifier on three discriminant features: length, G+C content, and the CpG ratio (obversed/expected). The information about CpG_Promoter is available at ftp://cshl.org/pub/science/mzhanglab/ioshikhes/. It should be mentioned that, like CpGPlot, PIPmaker can also display CpG islands in a large genome.

10.4.2 TSS Prediction Based on TF Site Scan

As TF sites are over-represented in the promoter region, it is natural to seek a prediction program based on putative TF site density. PROMOTERSCAN is one such program developed by Prestridge (1995). It was based on the study of three datasets: a TF database (TFD: Ghosh 1993), a promoter database (EPD: Bucher and Trifonov 1986), and a nonpromoter set constructed from protein and RNA gene sequences. Density of all putative TF sites is calculated separately for promoter and nonpromoter sequences (within a 250-bp window upstream of TSS for promoters) and use the radio $R = D_p/D_n$ of the two densities as the scoring function supplemented with a TATA-matrix score (Bucher 1990) when scanning a test sequence using the same window. A web server for the program is available at http:// molbio.cbs.umn.edu/software/software.html.

A very similar, albeit statistically more sophisticated, approach was taken by Kondrakhin et al. (1995) and implemented in AUTOGENE.

10.4.3 TSS Prediction Based on *k*-tuples

Methods based on putative TF sites do have severe limitations: important context effect may be overlooked, the majority of putative sites are false positives, site matrices are scoring function/cut-off dependent, the database is biased by the limited number of known samples, and so on. A statistical learning approach without using any putative TF site information has become the attractive alternative. PromFind (Hutchinson 1996) is based on the difference in 6-mer frequencies between promoters, coding regions and noncoding regions downstream of the first coding exon. Among all sites in an input sequence where the promoter versus coding region discriminant exceeds a certain threshold, the site where the promoter versus noncoding region discriminant reaches its maximum (over the input sequence) is taken as a promoter. But this "content" approach would lose all the positional information. TSSG and TSSW (Solovyev and Salamov 1997) both use LDA (linear discriminant analysis) to combine (1) a TATA score, (2) triplet preferences around TSS, (3) 6-tuple score in three non-overlapping windows of 100 bp upstream TSS, and (4) putative binding site scores. TSSG is based on TFD (Ghosh 1993); TSSW is based on TRANSFAC (Wingender et al. 1996). They are available at http://dot.imgen.bcm.tmc.edu:9331/gene-finder/gf.html. Fickett and Hatzigeorgious (1997) had evaluated several earlier promoter prediction algorithms, including a Markov model–based algorithm (Audic and Claverie 1997). TSSW appeared to be ranked as one of the best, with a sensitivity of 42 percent and specificity of 1 false positive per 800 bp.

Using positional dependent 5-tuple measures, a QDA method for core promoter prediction was implemented in CorePromoter (Zhang 1998). Statistical tests indicated that when given a 2 KB upstream region, CorePromoter was able to localize the TSS to a 100-bp interval approximately 60 percent of the time. The ultility of CorePromoter and CpG_Promoter was recently demonstrated (Zhang 2000) in the re-analysis of human chromosome 22 genes in conjunction with our internal exon finder MZEF (Zhang 1997) and the terminal exon finder JTEF (Davuluri et al. 2000a).

A recent algorithm PromoterInspector (Scherf et al. 2000) is based on libraries of degenerate words extracted from training sequences by an unsupervised learning approach. It consists of three classifiers that discriminate promoter from intron, exon, and 3′-UTR separately and predicts a promoter when all three classifiers agree. Their test showed that 43 percent of the predictions can be expected to be true positives, whereas 43 percent of the annotated TSS were predicted correctly. PromoterInspector is available at http://genomatix.gsf.de/cgi-bi/promoterinspector/promoterinspector.pl.

In principle, TSS may also be predicted by discriminant analysis of 5′UTRs. With more full-length cDNAs becoming available, 5′UTR features will become as important as promoter features in TSS prediction (Davuluri et al. 2000b).

References

Aparicio, S., Morrison, A., Gould, A., Gilthorpe, J., Chaudhuri, C., Rigby, P., Krumlauf, R., and Brenner, S. (1995). Detecting conserved regulatory elements with the model genome of the Japanese puffer fish, *Fugu rubripes*. *Proc. Natl. Acad. Sci. USA* 92: 1684–1688.

Audic, S., and Claverie, J.-M. (1997). Detection of eukaryotic promoters using Markov transition matrices. *Comput. Chem.* 21: 223–227.

Bailey, T. L., and Elkan, C. P. (1994). Fitting a mixture model by expectation maximization to discover motifs in biopolymers. *Intell. Sys. Mol. Biol.* 2: 28–36.

Bailey, T. L., and Elkan, C. P. (1995). The value of prior knowledge in discoverin motifs with MEME. *Proc. 3rd Int. Conf. Intell. Sys. for Mol. Biol.* 21–29, Menlo Park, Calif.: AAAI Press.

Bailey, T. L., and Gribskov, M. (1996). The megaprior heuristic for discovering protein sequence patterns. *Proc. 4th Int. Conf. Intell. Sys. for Mol. Biol.* 5–24, Menlo Park, Calif.: AAAI Press.

Bailey, T. L., and Gribskov, M. (1998). Methods and statistics for combining motif match scores. *J. Comput. Biol.* 5: 211–221.

Baldi, P., and Brunak, S. (1998). *Bioinformatics: The machine learning approach*. Cambridge, Mass.: MIT Press.

Berg, O. G., and von Hippel, P. H. (1987). Selection of DNA binding sites by regulatory proteins. Statistical-mechanical theory and application to operators and promoters. *J. Mol. Biol.* 193: 723–750.

Bird, A. P. (1980). DNA methylation and the frequency of CpG in animal DNA. *Nucl. Acids Res.* 8: 1499–1504.

Bird, A. (1999). DNA methylation de novo. *Science* 286: 2287–2288.

Blackwood, E. M., and Kadonaga, J. T. (1998). Going the distance: A current view of enhancer action. *Science* 281: 61–63.

Brazma, A., Jonassen, I., Eidhammer, I., and Gilbert, D. (1998). Approaches to the automatic discovery of patterns in biosequences. *J. Comput. Biol.* 5: 279–305.

Bucher, P. (1990). Weight matrix descriptions of four eukaryotic RNA polymerase II promoter elements derived from 502 unrelated promoter sequences. *J. Mol. Biol.* 212: 563–578.

Bucher, P. (1999). Regulatory elements and expression profiles. *Curr. Opin. Struct. Biol.* 9: 400–407.

Bucher, P., and Trifonov, E. N. (1996). Copilation and analysis of eukaryotic PolII promoter sequences. *Nucl. Acids Res.* 14: 10009–10026.

Carey, M., and Smale, S. T. (1999). *Transcriptional Regulation in Eukaryotes*. Cold Spring Harbor, N.Y.: Cold Spring Harbor Laboratory Press.

Claverie, J.-M., and Audic, S. (1996). The statistical significance of nucleotide position-weight matrix matches. *CABIOS* 12: 431–439.

Claverie, J.-M., and Sauvaget, I. (1985). Assessing the biological significance of primary structure consensus patterns using sequence databanks. I. Heat-shock and glucocorticoid control elements in eukaryotic promoters. *CABIOS* 2: 95–104.

Cross, S. H., and Bird, A. P. (1995). CpG islands and genes. *Curr. Opin. Genet. Dev.* 5: 309–314.

Davuluri, R., Suzuki, Y., Sugano, S., and Zhang, M. Q. (2000a). CART classification of human 5′UTR sequences. Genome Res. 10: 1807–1816.

Davuluri, R., Tabaska, J., and Zhang, M. Q. (2000b). A novel 3′-Terminal exon recognition algorithm. Result was presented at CSHL Computational Biology Workshop Sept. 1999.

Delcher, A. L., Kasif, S., Fleischmann, R. D., Peterson, J., White, O., and Salzberg, S. L. (1999). Alignment of whole genomes. *Nucl. Acids Res.* 27: 2369–2376.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *J. R. Stat. Soc. Series* B 39: 1–38.

DeRisi, J. L., Iyer, V. R., and Brown, P. O. (1997). Exploring the metabolic and genetic control of gene expression on a genomic scale. *Science* 278: 680–686.

Durbin, R., Eddy, S. R., Krogh, A., and Mitchison, G. (1998). *Biological Sequence Analysis*. Cambridge, Eng.: Cambridge University Press.

Duret, L., and Bucher, P. (1997). Searching for regulatory elements in human noncoding sequences. *Curr. Opin. Struct. Biol.* 7: 399–406.

Fickett, J. W. (1996). Coordinate positioning of MEF2 and myogenin binding sites. *Gene* 172: GC19–GC32.

Fickett, J. W., and Wasserman, W. W. (2000). Discovery and modeling of transcriptional regulatory regions. *Curr. Opin. Biotech.* 11: 19–24.

French, K., Quandt, K., and Werner, T. (1997a). Finding protein-binding sites in DNA sequences: The next generation. *Trends. Biochem. Sci.* 22: 103–104.

French, K., Quandt, K., and Werner, T. (1997b). Software for the analysis of DNA sequence elements of transcription. *CABIOS* 13: 89–97.

French, K., Quandt, K., and Werner, T. (1998). Muscle actin genes: A first step towards computational classification of tissue specific promoters. *Silico Biol.* 1: 29–38.

Fukunaga, K. (1990). *Introduction to statistical pattern recognition*. 2nd ed. New York: Academic Press.

Gardiner-Garden, M., and Frommer, M. (1987). CpG islands in vertebrate genomes. *J. Mol. Biol.* 196: 261–276.

Ghosh, D. (1993). Status of the transcription factors database (TFD). *Nucl. Acids Res.* 21: 3117–3118.

Gumucio, D. L., Shelton, D. A., Zhu, W., Millinoff, D., Gray, T., Bock, J. H., Slightom, J. L., and Goodman, M. (1996). Evolutionary strategies for the elucidation of cis and trans factors that regulate the developmental switching programs of the β-like globin genes. *Mol. Phylogenet. Evol.* 5: 18–32.

Heinemayer, T., Wingender, E., Reuter, I., Hermjakob, H., Kel, A. E., Kel, O. V., Ignatieva, E. V., Ananko, E. A., Podkolodnaya, O. A., Kolpakov, F. A., Podkolodny, N. L., and Kolchanov, N. A. (1998). Databases on transcriptional regulation: TRANSFAC, TRRD and COMPEL. *Nucl. Acids Res.* 26: 362–367.

Hertz, G. Z., Hartzell, G. W. 3rd, and Stormo, G. D. (1990). Identification of consensus patterns in unaligned DNA sequences known to be functionally related. *Comput. Appl. Biosci.* 6: 81–92.

Hertz, G. Z., and Stormo, G. D. (2000). Identifying DNA and protein patterns with statistically significant alignments of multiple sequences. *Bioinformatics* 15: 563–577.

Holmes, M. C., and Tjian, R. (2000). Promoter-selective properties of the TBP-related factor TRF1. *Science* 288: 867–870.

Hosmer, D. W., and Lemeshow, S. (1989). *Applied Logistic Regression*. New York: John Wiley & Sons.

Hughes, J. D., Estep, P. W., Tavazoie, S., and Church, G. M. (2000). Computational identification of cis-regulatory elements associated with groups of functionally related genes in Saccharomyces cerevisiae. *J. Mol. Biol.* 296: 1205–1214.

Hutchinson, G. B. (1996). The prediction of vertebrate promoter regions using differential hexamer frequency analysis. *CABIOS* 12: 391–398.

Ioshikhes, I. P., Trifonov, E. N., and Zhang, M. Q. (1999). Periodical distribution of transcription factor sites in promoter regions and connection with chromatin structure. *Proc. Natl. Acad. Sci. USA* 96: 2891–2895.

Ioshikhes, I. P., and Zhang, M. Q. (2000). Large-scale human promoter mapping using CpG islands. *Nature Genet.* 26: 61–63.

Jareborg, N., Birney, E., and Durbin, R. (1999). Comparative analysis of noncoding regions of 77 orthologous mouse and human gene pairs. *Genome Res.* 9: 815–824.

Kondrakhin, Y. V., Kel, A. E., Kolchanov, N. A., Romashchenko, A. G., and Milanesi, L. (1995). Eukaryotic promoter recognition by binding sites for transcription factors. *CABIOS* 11: 477–488.

Larsen, F., Gundersen, R., Lopez, R., and Prydz, H. (1992). CpG islands as gene markers in the human genome. *Genomics* 13: 1095–1107.

Lawrence, C. E., Altschul, S. F., Bogouski, M. S., Liu, J. S., Neuwald, A. F., and Wooten, J. C. (1993). Detecting subtle sequence signals: A Gibbs sampler strategy for multiple alignment. *Science* 262: 208–214.

Lawrence, C. E., and Reilly, A. A. (1990). An expectation maximization (EM) algorithm for the identification of common sites in unaligned biopolymer sequences. *Proteins* 7: 41–51.

Lockhart, D. J., Dong, H., Byme, M. C., Follettie, M. T., Gallo, M. V., Chee, M. S., Mittmann, M., Wang, C., Kobayashi, M., Horton, H., and Brown, E. L. (1996). Expression monitoring by hybridization to high-density oligonucleotide array. *Nat. Biotech.* 14: 1675–1680.

Loots, G. G., Locksley, R. M., Blankespoor, C. M., Wang, Z. E., Miller, W., Rubin, E. M., and Frazer, K. A. (2000). Identification of a coordinate regulator of interleukins 4, 13, and 5 by cross-species sequence comparisons. *Science* 288: 136–140.

Marsan, L., and Sagot, M.-F. (2000). Extracting structured motifs using a suffix tree — algorithms and application to promoter consensus identification. *RECOM'2000.*

McGuire, A. M., Hughes, J. D., and Church, G. M. (2000). Conservation DNA regulatory motifs and discovery of new motifs in microbial genomes. *Genome Res.* 10: 744–757.

Neuwald, A. F., Liu, J. S., and Lawrence, C. E. (1995). Gibbs motif sampling: Detection of bacterial outer membrane repeats. *Protein Science* 4: 1618–1632.

Novina, C. D., and Roy, A. L. (1996). Core promoter and transcriptional control. *Trends Genet.* 12: 351–355.

Parvin, M. J., and Young, R. A. (1998). Regulatory targets in the RNA polymerase II homoenzyme. *Curr. Opin. Genet. Dev.* 8: 565–570.

Pedersen, A. G., Baldi, P., Chauvin, Y., and Brunak, S. (1999). The biology of eukaryotic promoter prediction — a review. *Comp. & Chem.* 23: 191–207.

Pesole, G., Attimonelli, M., and Saccone, C. (1996). Linguistic analysis of nucleotide sequences: Algorithms for pattern recognition and analysis of codon strategy. *Met. Enz.* 266: 281–294.

Pesole, G., Prunella, N., Liuni, S., Attimonelli, M., and Saccone, C. (1992). WORDUP: An efficient algorithm for discovering statistically significant patterns in DNA sequences. *Nucl. Acids Res.* 20: 2871–2875.

Prestridge, D. S. (1995). Predicting PolII promoter sequences using transcription factor binding sites. *J. Mol. Biol.* 249: 923–932.

Roth, F. P., Hughes, J. D., Estep, P. W., and Church, G. M. (1998). Finding DNA regulatory motifs within unaligned noncoding sequences clustered by whole genome mRNA quantitation. *Nat. Biotech.* 16: 939–945.

Roulet, E., Bucher, P., Schneider, R., Wingender, E., Dusserre, Y., Werner, T., and Mermod, N. (2000). Experimental analysis and computer prediction of CTF/NFI transcription factor DNA binding sites. *J. Mol. Biol.* 297: 833–848.

Schena, M., Shalon, D., Davis, R. W., and Brown, P. O. (1995). Quatitative monitoring of gene expression patterns with a complementary DNA microarray. *Science* 270: 467–470.

Scherf, M., Klingenhoff, A., and Werner, T. (2000). Highly specific localization of promoter regions in large genomic sequences by Promoter Inspector: A novel context analysis approach. *J. Mol. Biol.* 297: 599–606.

Schneider, T. D., Stormo, G. D., Gold, L., and Ehrenfeucht, A. (1986). Information content of binding sites on nucleotide sequences. *J. Mol. Biol.* 188: 415–431.

Schwartz, S., Zhang, Z., Frazer, K. A., Smit, A., Riemer, C., Bouck, J., Gibbs, R., Hardison, R., and Miller, W. (2000). PipMaker a web server for aligning two genomic DNA sequences. *Genome Res.* 10: 577–586.

Smale, S. T. (1994). Core promoter architecture for eukaryotic protein-coding genes. In *Transcription: Mechanisms and Regulation*, Conaway, R. C. and Conaway, J. W. eds., 63 81. New York: Raven Press.

Solovyev, V., and Salamov, A. (1997). The Gene-Finder computer tools for analysis of human and model organism genome sequences. In *Proc. 5th Int. Conf. Intell. Sys. for Mol. Biol.* Gaasterland, T., Karp, P., Karplus, K., Ouzounis, C., Sander, C., and Valencia, A., 294 302. Menlo Park, Calif.: AAAI Press.

Spellman, P. T., Sherlock, G., Zhang, M. Q., Iyer, V. R., Anders, K., Eisen, M. B., Brown, P. O., Botstein, D., and Futcher, B. (1998). Comprehensive identification of cell cycle-regulated genes of the yeast *Saccharomyces cerevisiae* by microarray hybridization. *Mol. Biol. Cell.* 9: 3273 3297.

Staden, R. (1989). Methods for calculating the probabilities of finding patterns in sequences. *CABIOS* 5: 89–96.

Stojanovich, N., Florea, L., Riemer, C., Gumucio, D., Slightom, J., Goodman, M., Miller, W., and Hardison, R. (1999). Comparison of five methods for finding conserved sequences in multiple alignments of gene regulatory regions. *Nucl. Acids Res.* 27: 3899 3910.

Stormo, G. D. (2000). DNA binding sites: Representation and discovery. *Bioinformatics* 16: 16 23.

Stormo, G. D., and Hartzell, G. W. (1989). Identifying protein-binding sites from unaligned DNA fragments. *Proc. Natl. Acad. Sci. USA* 86: 1183–1187.

Thanos, D., and Maniatis, T. (1995). Virus induction of human IFN β gene expression requires the assembly of an enhanceosome. *Cell* 29: 1091 1100.

Vanet, A., Marsan, L., and Sagot, M.-F. (1999). Promoter sequences and algorithmical methods for identifying them. *Res. Microbiol.* 150: 779–799.

Van Helden, J., Andre, B., and Collado-Vides, J. (1998). Extracting regulatory sites from the upstream region of yeast genes by computational analysis of oligonucleotide frequencies. *J. Mol. Biol.* 281: 827 842.

Wasserman, W. W., and Fickett, J. W. (1998). Identification of regulatory regions which confer muscle-specific gene expression. *J. Mol. Biol.* 278: 167 181.

Werner, T. (1999). Models for prediction and recognition of eukaryotic promoters. *Mamm. Genet.* 10: 168 175.

Wingender, E. P., Dietze, P., Karas, H., and Knueppel, R. (1996). TRANSFAC: A database on transcription factors and their DNA binding sites. *Nucl. Acids Res.* 24: 238 241.

Zhang, M. Q. (1997). Identification of protein coding regions in the human genome based on quadratic discriminant analysis. *Proc. Natl. Acad. Sci. USA* 94: 565–568.

Zhang, M. Q. (1998). Identification of human gene core promoters in silico. *Genome Research* 8: 319 326.

Zhang, M. Q. (1999a). Promoter analysis of co-regulated genes in the yeast genome. *Comp. & Chem.* 23: 233 250.

Zhang, M. Q. (1999b). Large-scale gene expression data analysis: A new challenge to computational biologists. *Genome Research* 9: 681–688.

Zhang, M. Q. (2000). Discriminant analysis and its application in DNA sequence motif recognition. Presented at the Gene-finding Workshop, June 2000, EBI, Cambridge, UK. To appear in *Briefings in Bioinformatics*.

Zhang, M. Q., and Marr, T. G. (1993). A weight array method for splicing signal analysis. *CABIOS* 9: 499–509.

Zhu, J., and Zhang, M. Q. (1999). SCPD: A promoter database of yeast *Saccharomyces cerevisiae*. *Bioinformatics* 15: 607 611.

Zhu, J., and Zhang, M. Q. (2000). Cluster, function and promoter: Analysis of yeast expression array. In *Proceedings of Pacific Symposium on Biocomputing 2000*. Altman, R. B., et al., eds. 5: 476 487.

11 Algorithmic Approaches to Clustering Gene Expression Data

Ron Shamir and Roded Sharan

11.1 Introduction

Technologies for generating high-density arrays of cDNAs and oligonucleotides are developing rapidly, changing the landscape of biological and biomedical research. They enable, for the first time, a global, simultaneous view of the transcription levels of many thousands of genes, when the cell undergoes specific conditions or processes. For several organisms that have had their genomes completely sequenced, the full set of genes can already be monitored this way today. The potential of such technologies is tremendous. The information obtained by monitoring gene expression levels in different developmental stages, tissue types, clinical conditions, and different organisms can help the understanding of gene function and gene networks, and assist in the diagnostic of disease conditions and effects of medical treatments. Undoubtedly, other applications will emerge in coming years.

A key step in the analysis of gene expression data is the identification of groups of genes that manifest similar expression patterns. This translates to the algorithmic problem of clustering gene expression data. A clustering problem consists of elements and a characteristic vector for each element. A measure of similarity is defined between pairs of such vectors. (In gene expression, elements are usually genes, the vector of each gene contains its expression levels under each of the monitored conditions, and similarity can be measured, for example, by the correlation coefficient between vectors.) The goal is to partition the elements into subsets, which are called *clusters*, so that two criteria are satisfied: *homogeneity*—elements in the same cluster are highly similar to each other; and *separation*—elements from different clusters have low similarity to each other.

In this chapter we describe some of the main algorithmic approaches to clustering gene expression data. Clustering is a fundamental problem that has numerous other applications in biology as well as in many other disciplines. It also has a very rich literature, going back at least a century, and according to some authors, all the way to Aristo. Any such review is thus necessarily incomplete, and reflects the background, taste, and biases of the authors.

11.2 Biological Background

In this section we outline three technologies that generate large-scale gene expression data. All three are based on performing of a large number of hybridization experiments in parallel on high-density arrays (a.k.a. "DNA chips") between probes and

targets. They differ in the nature of the probes and the targets and in other technological aspects, which raise different computational issues in analyzing the data. For more on the technologies and their applications see, for example, Marshall and Hodgson (1998), Ramsay (1998), Eisen and Brown (1999), Chipping (1999), Lockhart and Winzeler (2000).

11.2.1 cDNA Microarrays

cDNA microarrays (Schena et al. 1996; Schena 1996; Marshall and Hodgson 1998; Ramsay 1998) are microscopic arrays that contain large sets of cDNA sequences immobilized on a solid substrate. In an array experiment, many gene-specific cDNAs are spotted on a single matrix. The matrix is then simultaneously probed with fluorescently tagged cDNA representations of total RNA pools from test and reference cells, allowing one to determine the relative amount of transcript present in the pool by the type of fluorescent signal generated. Current technology can generate arrays with over ten thousand cDNAs per square centimeter.

cDNA microarrays are produced by spotting PCR products of approximately 0.6–2.4 KB, representing specific genes onto a matrix. The spotted cDNAs are usually chosen from appropriate databases, such as GenBank (Benson et al. 1999) and UniGene (Schuler 1997). Additionally, cDNAs from any library of interest (whose sequences may be known or unknown) can be used. Each array element is generated by the deposition of a few nanoliters of purified PCR product. Printing is carried out by a robot that spots a sample of each gene product onto a number of matrices in a serial operation.

To maximize the reliability and precision with which quantitative differences in the abundance of each RNA species are detected, one directly compares two samples (test and reference) by labeling them with spectrally distinct fluorescent dyes and mixing the two probes for simultaneous hybridization to one array. The relative representation of a gene in the two samples is assayed by measuring the ratio of the (normalized) fluorescent intensities of the two dyes at the target element. Cy3-dUTP and Cy5-dUTP are frequently used as the fluorescent labels. For the comparison of multiple samples, such as in time-course experiments, one often uses the same reference sample with each of the test samples.

11.2.2 Oligonucleotide Microarrays

In oligonucleotide microarrays (Fodor et al. 1993; Lipshutz et al. 2000; Harrington et al. 2000), each spot on the array contains a short synthetic oligonucleotide (oligo), typically 20–30 bases long. The oligos are designed based on the knowledge of the DNA (or EST) target sequences, to ensure high affinity and specificity of each oligo to

a particular target gene. Moreover, they should not be near-complementary to other RNAs that may be highly abundant in the sample (e.g., rRNAs, tRNAs, alu-like sequences, etc.).

One of the leading approaches to the construction of high-density DNA probe arrays employs photolithography and solid-phase DNA synthesis. First, synthetic linkers, modified with a photochemically removable protecting groups are attached to a glass substrate. At each phase, light is directed through a photolithographic mask to specific areas on the surface to produce localized deprotection. Specific Hydroxyl-protected deoxynucleosides are incubated with the surface, and chemical coupling occurs at those sites that have been illuminated. Current technology allows for approximately 300,000 oligos to be synthesized on a 1.28 × 1.28 cm array. Key to this approach is the use of multiple distinct oligonucleotides designed to hybridize to different regions of the same RNA. This use of multiple detectors greatly improves signal-to-noise ratio and accuracy of RNA quantitation, and reduces the rate of false-positives and miscalls.

An additional level of redundancy comes from the use of mismatch control probes that are identical to their perfect match partners except for a single base difference in a central position. These probes act as specificity controls. They allow the direct subtraction of both background and cross-hybridization signals, and allow discrimination between "real" signals and those due to non-specific or semi-specific hybridizations.

11.2.3 Oligonucleotide Fingerprinting

Historically, the Oligonucleotide Fingerprinting (ONF) method preceded the other two (Lennon and Lehrach 1991; Drmanac et al. 1991; Vicentic and Gemmell 1992; Drmanac and Drmanac 1994; Drmanac et al. 1996; Meier-Ewert et al. 1995; Milosavljevic et al. 1995). It was initially proposed in the context of sequencing by hybridization, as an alternative to DNA sequencing. Although that approach to sequencing is currently not competitive, ONF has found other good applications, including gene expression. It can be used to extract gene expression information about a cDNA library from a specific tissue under analysis, without prior knowledge of the genes involved. Conceptually, it takes the "reverse" approach to that of the oligo microarrays: The target is on the array, and the oligos are "in the air."

The ONF method is based on spotting the cDNAs on high density nylon membranes (about 31,000 different cDNA can be spotted currently in duplicates on one filter. See (Drmanac et al. 1996). A large quantity of a short synthetic oligo, typically 7–12 bases long, radioactively labeled, is put in touch with the membrane in proper

conditions. The oligos hybridize to those cDNAs that contain a DNA sequence complementary to that of the oligo. By inspecting the filter one can detect to which of the cDNAs the oligo hybridized. Hence, ideally, the result of such an experiment is one 1/0 bit for each of the cDNAs.

The experiment is repeated with p different oligos, giving rise to a p-long vector for each cDNA spot, indicating which of the (complements of) oligo sequences are contained in each cDNA. This *fingerprint* vector, similar to a barcode, identifies the cDNA. Thus, distinct spots of cDNAs originating from the same gene should have similar fingerprints. By clustering these fingerprints, one can identify cDNAs originating from the same gene. The larger that number, the higher the expression level of the corresponding gene. Gene identification can subsequently be obtained by sample sequencing or by comparison of average cluster fingerprints to a sequence database (Poustka et al. 1999).

Because of the short oligos used, the hybridization information is rather noisy, but this can be compensated by using a longer fingerprint. The method is probably less efficient than the other two methods, which measure abundance directly in a single spot. However, it has the advantage of applicability to species with unknown genomes, which oligo microarrays cannot handle, and it requires relatively lower mRNA quantities than cDNA microarrays.

11.3 Mathematical Formulations and Background

Let $N = \{e_1, \ldots, e_n\}$ be a set of n elements, and let $\mathscr{C} = (C_1, \ldots, C_l)$ be a *partition* of N into subsets. That is, the subsets are disjoint and their union is N. Each subset is called a *cluster*, and $\mathscr{C}$ is called a *clustering solution*, or simply a *clustering*. Two elements, e_i and e_j, are called *mates with respect to* $\mathscr{C}$ if they are members of the same cluster in $\mathscr{C}$. In the gene expression context, the elements are the genes and we often assume that there exists some correct partition of the genes into "true" clusters. When $\mathscr{C}$ is the true clustering of N, elements that belong to the same true cluster are simply called mates.

The input data for a clustering problem is typically given in one of two forms: (1) *Fingerprint data*—each element is associated with a real-valued vector, called its *fingerprint*, or *pattern*, which contains p measurements on the element, such as expression levels of an mRNA at different conditions (cf. Eisen and Brown 1999), or hybridization intensities of a cDNA with different oligos (cf. Lennon and Lehrach 1991). (2) *Similarity data*—pairwise similarity values between elements. These values can be computed from fingerprint data, such as by correlation between vectors. Alter-

natively, the data can represent pairwise dissimilarity, for example, by computing distances. Fingerprints contain more information than similarity, but the latter is completely generic and can be used to represent the input to clustering in any application. (Note that there is also a practical consideration regarding the presentation: the fingerprint matrix is of order $n \times p$, whereas the similarity matrix is of order $n \times n$, and in gene expression applications often $n \gg p$.)

The goal in a clustering problem is to partition the set of elements N into homogeneous and well-separated clusters. That is, we require that elements from the same cluster will be highly similar to each other, whereas elements from different clusters will have a low similarity to each other. Note that this formulation does not define a single optimization problem: homogeneity and separation can be defined in various ways, leading to a variety of optimization problems. Note also that even when the homogeneity and separation are precisely defined, those are two objectives that are typically conflicting: the higher the homogeneity, the lower the separation, and vice versa. The lack of a single objective agreed upon by the community is inherent in the clustering problem, a point we will return to in the sequel.

Clustering problems and algorithms are often represented in graph-theoretic terms. We therefore include some basic definitions on graphs. We refer the readers to Golumbic (1980), and Even (1979) for more background and terminology on graphs.

Let $G = (V, E)$ be a weighted graph. We denote the vertex set of G also by $V(G)$. For a subset $R \subseteq V$, *the subgraph induced by* R, denoted G_R, is obtained from G by deleting all vertices not in R and the edges incident on them. That is, $G_R = (R, E_R)$ where $E_R = \{(i, j) \in E \mid i, j \in R\}$. For a vertex $v \in V$, define the *weight* of v to be the sum of weights of the edges incident on v. A *cut* C in G is a subset of its edges, whose removal disconnects G. The *weight* of C is the sum of the weights of its edges. A *minimum weight cut* is a cut in G with minimum weight. In case of non-negative edge weights, a minimum weight cut C partitions the vertices of G into two disjoint non-empty subsets $A, B \subset V$, $A \cup B = V$, such that $E \cap \{(u, v) : u \in A, v \in B\} = C$. For a pair of vertices $u, v \in V$, the *distance* between u and v is the length of the shortest path that connects them. The *diameter* of G is the maximum distance between a pair of vertices in G. For an example of these definitions, see figure 11.1.

For a set of elements $K \subseteq N$, we define the *fingerprint* or *centroid* of K to be the mean vector of the fingerprints of the members of K. For two fingerprints x and y, we denote their similarity by $S(x, y)$ and their dissimilarity by $d(x, y)$. A *similarity graph* is a weighted graph in which vertices correspond to elements and edge weights are derived from the similarity values between the corresponding elements. Hence, the similarity graph is just another representation of the similarity matrix.

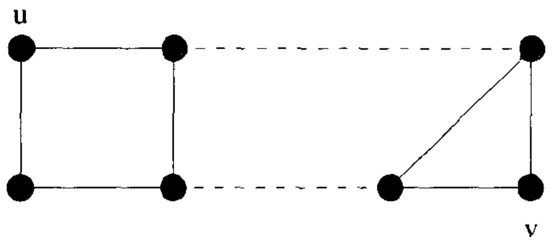


Figure 11.1
A graph and a corresponding minimum weight cut, assuming that all edge weights are 1. Minimum cut edges are denoted by broken lines. The length of the shortest path between u and v is 3, which is also the diameter of the graph.

An alternative formulation of the clustering problem is hierarchical: rather than asking for a single partition of the elements, one seeks an iterated partition: A *dendogram* is a rooted weighted tree, with leaves corresponding to elements. Each edge defines the cluster of elements contained in the subtree below that edge. The edge's weight (or length) reflects the dissimilarity between that cluster and the remaining elements. In this formulation the clustering solution is the dendogram, and each non-singleton cluster, corresponding to a rooted subtree, is split into subclusters. The determination of disjoint clusters is left to the judgment of the user. Typically, one tends to consider as genuine clusters elements of a subtree just below a connecting edge of high weight.

Irrespective of the representation of the clustering problem input, judicious preprocessing of the raw data is key to meaningful clustering. This preprocessing is application dependent and must be chosen in view of the expression technology used and the biological questions asked. The goal of preprocessing is to normalize the data and calculate the pairwise element (dis)similarity, if applicable. Common procedures for normalizing fingerprint data include transforming each fingerprint to have mean of 0 and variance of 1, a fixed norm, or a fixed maximum entry. Statistically based methods for data normalization have also been developed recently (cf. Kerr et al. 2000).

11.4 Algorithms

Several algorithmic techniques were previously used in clustering gene expression data, including hierarchical clustering (Eisen et al. 1998), self-organizing maps (Tamayo et al. 1999), and graph theoretic approaches (Hartuv et al. 2000; Ben-Dor et al. 1999; Sharan and Shamir 2000b). We describe these approaches in the sequel. For other approaches to clustering expression patterns, see Milosavljevic et al. (1995); Alon

1. Find a minimal entry $d_{i^*j^*}$ in D, and merge clusters i^* and j^*.

2. Modify D by deleting rows and columns i, j and adding a new row and column $i^* \cup j^*$, with their dissimilarities defined by:

$$d_{k,i^* \cup j^*} = d_{i^* \cup j^*,k} = \alpha_{i^*} d_{ki^*} + \alpha_{j^*} d_{kj^*} + \gamma |d_{ki^*} - d_{kj^*}|$$

3. If there is more than one cluster, then go to Step 1.

Figure 11.2
The agglomerative hierarchical clustering scheme.

et al. (1999); Getz et al. (2000b); and Heyer et al. (1999). Much more information and background on clustering is available (cf. Hartigan 1975; Everitt 1993; Mirkin 1996; Hansen and Jaumard 1997).

Several algorithms for clustering were developed by first designing a "clean" algorithm that has proven properties, either in terms of time complexity, or in terms of (deterministic or probabilistic) solution quality. Then a more efficient yet heuristic algorithm is developed based on the same idea. We shall describe here the heuristics used in practice, but refer also briefly to the properties of the theoretical algorithm that motivated them.

11.4.1 Hierarchical Clustering

Hierarchical clustering solutions are typically represented by a dendogram. Algorithms for generating such solutions often work either in a top-down manner, by repeatedly partitioning the set of elements, or in a bottom-up fashion. We shall describe here the latter. Such *agglomerative* hierarchical clustering algorithms are among the oldest and most popular clustering methods (Cormack 1971). They proceed from an initial partition into singleton clusters by successive merging of clusters until all elements belong to the same cluster. Each merging step corresponds to joining two clusters. The general scheme due to Lance and Williams (1967) is presented in figure 11.2. It is assumed that $D = (d_{ij})$ is the input dissimilarity matrix.

Common variants of this scheme are the following:

- *Single linkage*: $d_{k,i^* \cup j^*} = \min\{d_{ki^*}, d_{kj^*}\}$. Here $\alpha_{i^*} = \alpha_{j^*} = 1/2$ and $\gamma = -1/2$.
- *Complete linkage*: $d_{k,i^* \cup j^*} = \max\{d_{ki^*}, d_{kj^*}\}$. Here $\alpha_{i^*} = \alpha_{j^*} = 1/2$ and $\gamma = 1/2$.

1. Start with an arbitrary partition P of N into k clusters.

2. For each element i and cluster $j \neq P(i)$ let E_P^{ij} be the cost of a solution in which i is moved to cluster j. **If** $E_P^{i^*j^*} - min_{ij} E_P^{ij} < E_P$ **then** move i^* to cluster j^* and repeat Step 2. Otherwise **halt**.

Figure 11.3
The K-means algorithm.

- *Average linkage*: $d_{k,i^* \cup j^*} = n_{i^*} d_{ki^*}/(n_{i^*} + n_{j^*}) + n_{j^*} d_{kj^*}/(n_{i^*} + n_{j^*})$, where n_i denotes the number of elements in cluster i. Here $\alpha_{i^*} = n_{i^*}/(n_{i^*} + n_{j^*})$, $\alpha_{j^*} = n_{j^*}/(n_{i^*} + n_{j^*})$, and $\gamma = 0$.

Eisen et al. (1998) developed a clustering software package based on the average-linkage hierarchical clustering algorithm. The software package is called Cluster, and the accompanying visualization program is called TreeView. Both programs are available at http://rana.Stanford.EDU/software/. The gene similarity metric used is a form of correlation coefficient. The algorithm iteratively merges elements whose similarity value is the highest, as explained above. The output of the algorithm is a dendogram and an ordered fingerprint matrix. The rows in the matrix are permuted based on the dendogram, so that groups of genes with similar expression patterns are adjacent. The ordered matrix is represented graphically by coloring each cell according to its content. Cells with log ratios of 0 are colored black, increasingly positive log ratios with reds of increasing intensity, and increasingly negative log ratios with greens of increasing intensity.

11.4.2 K-Means

K-means (MacQueen 1965; Ball and Hall 1967) is another classical clustering algorithm. It assumes that the number of clusters k is known, and aims to minimize the distances between elements and the centroids of their assigned clusters. Let M be the $n \times m$ fingerprint matrix. For a partition P of the elements in $\{1, \ldots, n\}$, denote by $P(i)$ the cluster assigned to i, and by $c(j)$ the centroid of cluster j. Let $d(v_1, v_2)$ denote the Euclidean distance between the fingerprint vectors v_1 and v_2. K-means tries to find a partition P for which the error-function $E_P = \sum_{i=1}^{n} d(i, c(P(i)))$ is minimum.

Each iteration of K-means modifies the current partition by checking all possible modifications of the solution in which one element is moved to another cluster, and making a switch that reduces the error function. Figure 11.3 describes the most basic

1. Start with a set of sufficiently different elements as clusters.
2. **For** each remaining element i **do:**
   - For each cluster C s.t. $S(i, C) \geq \rho$ **do:**
     - add i to C.
     - **While** there exists a cluster C' s.t. $S(C, C') > \gamma$, merge C' into C.
   - **If** i was not added to any cluster **then** form a new cluster $\{i\}$.
3. Assign each element to the cluster to which it is most similar.

Figure 11.4
The algorithm of Herwig et al.

scheme. A more efficient variant moves in one iteration all elements that would benefit from a move: for each i simultaneously, if $min_j\ E_P^{ij} < E_P$, move i to the cluster j minimizing E_P^{ij}. This algorithm is very easy to implement and is used in many applications.

Another heuristic inspired by K-means was developed by Herwig et al. (1999) to cluster cDNA oligo-fingerprints. Unlike the regular K-means algorithm, this algorithm does not require a prespecified number of clusters. Instead, it uses two parameters: γ is the maximal admissible similarity of two distinct clusters, and ρ is the maximal admissible similarity between an element and a cluster different from its own cluster. (Similarity to a cluster is similarity to its centroid.) Elements are handled one at a time, added to sufficiently close clusters, or otherwise, form a new cluster. Whenever centroids become too close, their clusters are merged. Unlike the K-means algorithm, an element may be tentatively assigned to more than one cluster, and thus influence the location of several centroids to which it is sufficiently close. The algorithm is shown in figure 11.4. Here $S(i, C)$ is the similarity between element i and cluster C.

11.4.3 HCS and CLICK

The HCS (Hartuv et al. 2000; Hartuv and Shamir 1999) and CLICK (Sharan and Shamir 2000a, b) algorithms use a similar graph theoretic approach to clustering. The input data is represented as a similarity graph. The algorithm recursively partitions the current set of elements into two subsets. Before a partition, the algorithm

Form-Kernels(G):

If $V(G) = \{v\}$ **then** move v to the singleton set.

Else if G is a kernel **then** output $V(G)$.

Else

$(H, \bar{H}) \leftarrow$MinWeightCut(G).

Form-Kernels(H).

Form-Kernels($\bar{H}$).

Figure 11.5
The basic scheme of HCS and CLICK. Procedure MinWeightCut (G) computes a minimum weight cut of G and returns a partition of G into two subgraphs H and $\bar{H}$ according to this cut.

considers the subgraph induced by the current subset of elements. If the subgraph satisfies a stopping criterion, then it is declared a *kernel.* Otherwise, a minimum weight cut is computed in that subgraph, and the set is split into the two subsets separated by that cut. The output is a list of kernels that serve as a basis for the eventual clusters. This scheme is detailed in figure 11.5.

HCS and CLICK differ in the similarity graph they construct, their stopping criteria, and the postprocessing of the kernels. We describe each of the algorithms below.

HCS The HCS algorithm (Hartuv et al. 2000; Hartuv and Shamir 1999) builds from the input data an *unweighted* similarity graph G (each edge has weight 1 and each non-edge has weight 0) in which there is an edge between two vertices if and only if the similarity between their corresponding elements exceeds a predefined threshold.

The following notion is key to the algorithm: A *highly connected subgraph* (HCS) is an induced subgraph H of G, whose minimum cut value exceeds $|V(H)|/2$. That is, H remains connected if any $\lfloor |V(H)|/2 \rfloor$ of its edges are removed. The algorithm identifies highly connected subgraphs as kernels. Figure 11.6 demonstrates an application of the algorithm.

The HCS algorithm possesses several good properties for clustering (Hartuv and Shamir 1999): The diameter of each cluster it produces is at most two, and each cluster is at least half as dense as a clique. Both properties indicate strong cluster homogeneity. Inter-cluster separation is harder to prove, but it is argued that if errors are random, any nontrivial set split by the algorithm is unlikely to have diameter two unless the involved sets are small.

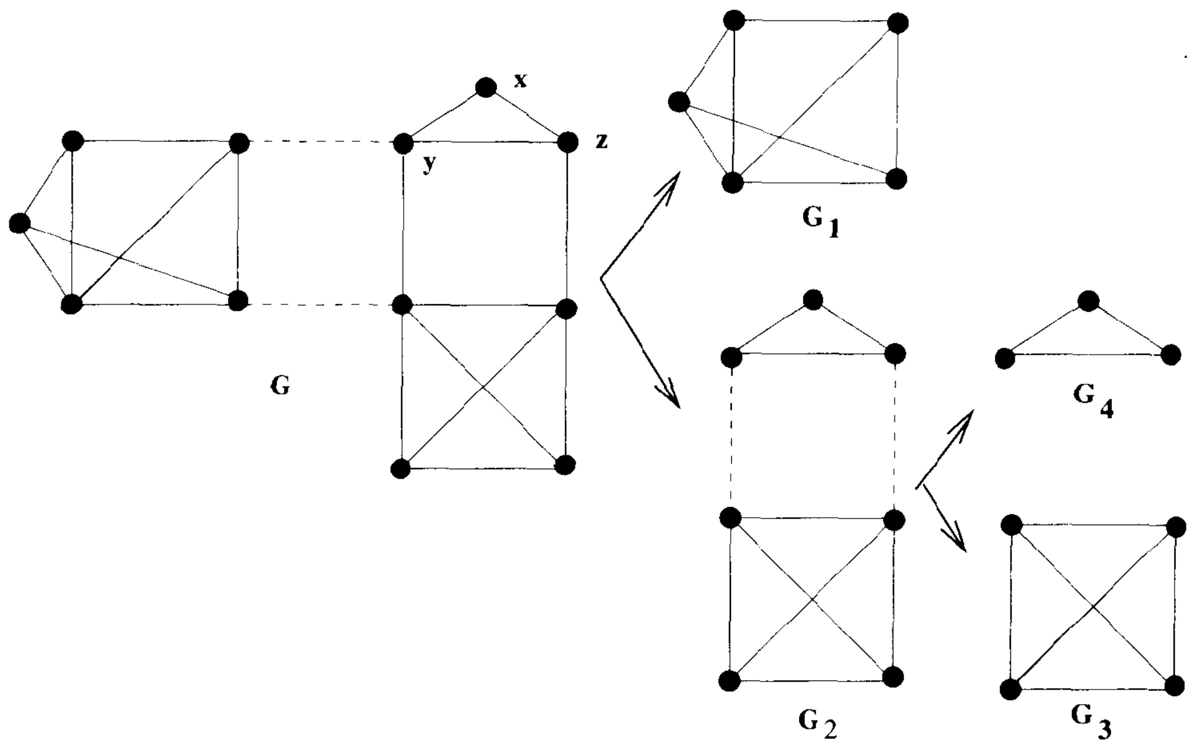


Figure 11.6
An example of applying the HCS algorithm to a graph. Minimum cut edges are denoted by broken lines.

To improve separation in practice, several heuristics are used to expand the kernels and speed up the algorithm:

ITERATED-HCS When the minimum cut value is obtained by several distinct cuts, the HCS algorithm chooses one arbitrarily. This process may break small clusters into singletons. (For example, a different choice of minimum cuts by the algorithm for the graph in figure 11.6 may split x from G_2 and eventually find the clusters G_1 and G_3, leaving x, y, z as singletons.) To overcome this, several (1–5) HCS iterations are carried out until no new cluster is found.

SINGLETONS ADOPTION Singletons can be "adopted" by clusters. For each singleton element x we compute the number of neighbors it has in each cluster and in the singletons set $\mathcal{S}$. If the maximum number of neighbors is sufficiently large, and is obtained by one of the clusters (rather than by $\mathcal{S}$), then x is added to that cluster. The process is repeated several times.

REMOVING LOW-DEGREE VERTICES When the similarity graph contains vertices with low degrees, one iteration of the minimum cut algorithm may simply separate a low degree vertex from the rest of the graph. This is computationally very expensive, not

informative in terms of the clustering, and may happen many times if the graph is large. Removing low-degree vertices from G eliminates such iterations, and significantly reduces the running time. The process is repeated with several thresholds on the degree. This simple procedure is very powerful for large problems.

CLICK The CLICK algorithm (CLuster Identification via Connectivity Kernels) (Sharan and Shamir 2000b), available at http://www.math.tau.ac.il/~rshamir/click/click.html, builds on a statistical model. The model gives probabilistic meaning to edge weights in the similarity graph and to the stopping criterion. The key probabilistic assumption of CLICK is that pairwise similarity values between elements are normally distributed: similarity values between mates are Normally distributed with mean μ_T and variance σ_T^2, and similarity values between nonmates are normally distributed with mean μ_F and variance σ_F^2, where $\mu_T > \mu_F$. This situation often holds on real data, and can be asymptotically justified (Sharan and Shamir 2000b).

The algorithm uses the values of μ_T, μ_F, σ_T, and σ_F, as well as the probability p_{mates}, that two randomly chosen elements are mates. These parameters can be computed directly from a known solution on a subset of the elements (which is often available in ONF experiments [Poustka et al. 1999]), or estimated using the EM algorithm, assuming the above probabilistic model for similarity values (see, e.g., Mirkin 1996, sec. 3.2.7).

Let $S = (S_{ij})$ be the input similarity matrix. Form a weighted similarity graph $G = (V, E)$, in which the weight w_{ij} of the edge (i, j) reflects the probability that i and j are mates, and is derived from the normal density function $f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-((x-\mu)^2/2\sigma^2)}$ and Bayes theorem:

$$w_{ij} = \ln \frac{Prob(i, j \text{ are mates} \mid S_{ij})}{Prob(i, j \text{ are non-mates} \mid S_{ij})}$$

$$= \ln \frac{p_{mates}\sigma_F}{(1 - p_{mates})\sigma_T} + \frac{(S_{ij} - \mu_F)^2}{2\sigma_F^2} - \frac{(S_{ij} - \mu_T)^2}{2\sigma_T^2}$$

CLICK uses the same basic scheme as HCS (see figure 11.5) to form kernels. The current subgraph is determined to be a kernel if the value of a minimum cut in it is positive. This is the case if and only if for every cut C in the current subgraph, the probability that it contains only edges between mates exceeds the probability that C contains only edges between nonmates.

The actual implementation omits from the graph all edges with values below some predefined non-negative threshold, computes the minimum cut in that simplified graph, and corrects the solution value for the missing edges.

$R \leftarrow N$.

While some change occurs **do**:

  Form-Kernels(G_R).

  Adoption($\mathcal{L}$, R).

Merge($\mathcal{L}$).

Adoption($\mathcal{L}$, R).

Figure 11.7
The CLICK algorithm. N is the complete set of elements (all the vertices in the similarity graph). Throughout the algorithm, $\mathcal{L}$ is the current list of kernels and R is the set of singletons. G_R is the subgraph of G induced by the vertex set R. Adoption($\mathcal{L}$, R) performs the iterative singletons adoption procedure. Merge($\mathcal{L}$) is the iterative merging procedure.

CLICK first produces kernels that form the basis of the eventual clusters. Subsequent processing includes singleton adoption, recursive clustering process on the set of remaining singletons, and an iterative *merging step*. The singletons adoption step is based on computing similarities between singletons' and clusters' fingerprints. The merging step iteratively merges two kernels whose fingerprint similarity is the highest, provided that this similarity exceeds a predefined threshold. The use of the fingerprints (rather than average similarity values) here is very powerful. Similar ideas were employed in Milosavljevic et al. (1995) and Hartuv et al. (2000). Finally, a last singleton adoption step is performed. The full algorithm is detailed in figure 11.7.

In order to reduce the running time of CLICK on very big instances, a screening heuristic is applied, similar to the low-degree heuristic of the HCS algorithm. Low-weight vertices are screened from large components in the following manner: First, the average vertex weight W of the component is computed, and is multiplied by a factor that is proportional to the logarithm of the size of the component. Denote the resulting threshold by W^*. Then vertices whose weight is below W^* are removed repeatedly, each time updating the weight of the remaining vertices, until the updated weight of every (remaining) vertex is greater than W^*. The removed vertices are marked as singletons and handled at a later stage.

11.4.4 CAST

Ben-Dor et al. (1999) developed a polynomial algorithm for finding true clustering with high probability, under the following stochastic model of the data. The underlying correct cluster structure is represented by a graph that is a disjoint union of

While there are unclustered elements **do**:

Pick an unclustered element to start a new cluster C.

Repeat ADD and REMOVE until no changes occur:

ADD: add an unclustered element v with maximum affinity to C if $a(v) > t|C|$.

REMOVE: remove an element u from C with minimum affinity if $a(u) \leq t|C|$.

Add C to the list of final clusters.

Figure 11.8
The CAST algorithm.

cliques, and errors are subsequently introduced in the graph by independently removing and adding edges between pairs of vertices with probability α. If all clusters are of size at least cn, for some constant $c > 0$, the algorithm solves the problem to a desired accuracy with high probability.

CAST uses as input the similarity matrix S. The *affinity* of an element v to a putative cluster C is $a(v) = \sum_{i \in C} S(i, v)$. The polynomial algorithm motivated the use of affinity to develop a faster heuristic called CAST (Clustering Affinity Search Technique) (Ben-Dor et al. 1999), which is implemented in the package BIOCLUST. The algorithm uses a single parameter t. Clusters are generated one by one. The next cluster is started with a single element, and elements are added or removed from the cluster if their relative affinity is larger or lower than t, respectively, until the process stabilizes. The algorithm is shown in figure 11.8.

An additional heuristic is employed at the end of the algorithm. A series of moving steps aims at a clustering in which the affinity of every element is higher to its assigned cluster than to any other cluster.

11.4.5 Self-organizing Maps

The self-organizing maps were developed by Kohonen (1997) as a method for fitting a number of ordered discrete reference vectors to the distribution of vectorial input samples. A self-organizing map (SOM) assumes that the number of clusters is known. Those clusters are organized as a set of nodes in a hypothetical "elastic network," with a simple neighborhood structure on the nodes, for example, a two-dimensional $k \times l$ grid. Each of these nodes is associated with a reference vector in $\mathscr{R}^n$. In the process of running the algorithm, the input vectors direct the movement of the reference vectors, so that an organization of the input vectors over the network emerges.

Arbitrarily set the reference vectors $f_1(v) \in R^n$ for each node v.

For $i = 1$ until no node location is changed by more than ϵ **do**:

Randomly pick a data point p.

Compute the node n_p with reference vector $f(n_p)$ closest to p.

Update all reference vectors: $f_{i+1}(n) = f_i(n) + \tau(n, n_p, i)[p - f_i(n)]$.

Assign each data point to the cluster with the closest reference vector.

Figure 11.9
The Self Organizing Map algorithm. The learning function $\tau(\cdot)$ monotonically decreases with $d(n, n_p)$ and with the iteration number i.

In the following we describe the SOM algorithm in the Euclidean space, and use $d(x, y)$ to denote the distance between points x and y.

The SOM process is iterative. Denote by $f_i(n)$ the position of node n at the i-th iteration. The initial positioning f_1 is random. The algorithm iteratively selects a random data point p, identifies the nearest reference node n_p, and updates the reference nodes according to a learning function $\tau(\cdot)$, where nodes closer to n_p are updated more. The updates also decrease with the iteration number. The algorithm is described in figure 11.9. The function $\tau(\cdot)$ represents the "stiffness" of the network. The intuition for this learning process is that the nodes that are close enough to p will "activate" each other to learn something from p.

Two popular choices for the learning function are:

• Neighborhood function: For each node n we denote by $N_i(n)$ the set of nodes within some distance from n. (These distances are with respect to the neighborhood structure in the network.) We then define $\tau(n, n_p, i) = 0$ if $n \notin N(n_p)$ and $\tau(n, n_p, i) = \alpha(i)$ otherwise. $\alpha(i)$ is called the learning rate, and it decreases with i.

• Gaussian function: $\tau(n, n_p, i) = \alpha(i) \cdot exp\left(-\frac{d(n, n_p)^2}{2\sigma^2(i)}\right)$, where $\alpha(i)$ and $\sigma(i)$ decrease with i.

For much more on self-organizing maps, see Kohonen (1997).

Tamayo et al. (1999) devised a gene expression clustering software, GeneCluster, which uses the SOM algorithm. The software is available at http://waldo.wi.mit.edu/MPR/. In their implementation, they incorporated a neighborhood learning function, for which $\alpha(i) = 0.02T/(T + 100i)$, where T is the maximum number of itera-

tions; and $N_i(n_p)$ contains all nodes whose distance to n_p is at most $p(i)$, where $p(i)$ decreases linearly with i, $p(0) = 3$.

GeneCluster accepts an input file of expression levels together with a two-dimensional grid geometry for the nodes. The number of grid points is the prescribed number of clusters. The resulting clusters are visualized by presenting for each cluster its average expression pattern with error-bars. Clusters are presented in their grid order, as clusters of close nodes tend to be similar.

Another implementation of SOM for clustering gene expression profiles was developed by Toronen et al. (1999).

11.5 Assessment of Solutions

A key question in the design and analysis of clustering techniques is how to evaluate solutions. We present in this section figures of merit for measuring the quality of a clustering solution. Different measures are applicable in different situations, depending on whether a partial true solution is known or not, and whether the input is fingerprint or similarity data. We describe below some of the applicable measures in each case. For other possible figures of merit, we refer the reader to Everitt (1993), Hansen and Jaumard (1997), and Yeung et al. (2000).

11.5.1 Assessment Given the True Solution

Suppose at first that the true solution is known, and we wish to compare it to a suggested solution. Any clustering solution can be represented by a binary $n \times n$ matrix C, in which $C_{ij} = 1$ if and only if i and j belong to the same cluster in that solution. Let T and C be the matrices for the true solution and the suggested solution, respectively. Let n_{kl}, $k, l = 0, 1$, denote the number of pairs (i, j) $(i \neq j)$ for which $T_{ij} = k$ and $C_{ij} = l$. Thus, n_{11} is the number of true mates that are also mates in the suggested solution, n_{00} is the number of nonmates correctly identified as such, whereas n_{01} and n_{10} count the disagreements between the true solution and the suggested one.

The *Minkowski measure* (see, e.g., Sokal 1977) is defined as

$$\sqrt{\frac{n_{01} + n_{10}}{n_{11} + n_{10}}}$$

Hence, it measures the proportion of disagreements to the total number of mates in the true solution. A perfect solution has score of 0, and the lower the score, the better the solution. The *Jaccard coefficient* (see, e.g., Everitt 1993) is the ratio

$$\frac{n_{11}}{n_{11} + n_{10} + n_{01}}$$

It is the proportion of correctly identified mates to the sum of the correctly identified mates plus the total number of disagreements. Hence, a perfect solution has score of 1, and the higher the score, the better the solution. This measure is a lower bound for both sensitivity $\left(\frac{n_{11}}{n_{11} + n_{10}}\right)$ and specificity $\left(\frac{n_{11}}{n_{11} - n_{01}}\right)$ of the solution.

Note that both measures do not (directly) involve the term n_{00}, as solution matrices tend to be sparse and this term would dominate the other three in good and bad solutions alike. When the true solution is known only for a subset $N^* \subset N$, the Minkowski and Jaccard measures can be computed on the submatrices corresponding to N^*. In some cases, such as for cDNA oligo-fingerprint data, we have the additional information that no element of N^* has a mate in $N \backslash N^*$. In such a case, the Minkowski and Jaccard measures are evaluated using all the pairs $\{(i, j) : i \in N^*, j \in N \cup N^*, i \neq j\}$.

11.5.2 Assessment When the True Solution is Unknown

When the true solution is not known, we evaluate the quality of a suggested solution by computing two figures of merit that measure its homogeneity and separation. For fingerprint data, homogeneity is evaluated by the average similarity between the fingerprint of an element and that of its cluster. Precisely, if $cl(u)$ is the cluster of u, $F(X)$ and $F(u)$ are the fingerprints of a cluster X and an element u, respectively, then

$$H_{Ave} = \frac{1}{|N|} \sum_{u \in N} S(F(u), F(cl(u)))$$

Separation is evaluated by the weighted average similarity between cluster fingerprints. That is, if the clusters are $X_1, \ldots X_t$, then

$$S_{Ave} = \frac{1}{\sum_{i \neq j} |X_i|\,|X_j|} \sum_{i \neq j} |X_i|\,|X_j| S(F(X_i), F(X_j))$$

Related measures that take a worst case instead of average case approach are minimum homogeneity: $H_{Min} = \min_{u \in N} S(F(u), F(cl(u)))$; and minimum separation: $S_{Max} = \max_{i \neq j} S(F(X_i), F(X_j))$. Hence, a solution improves if H_{Ave} or H_{Min} increase, and if S_{Ave} or S_{Max} decrease. In computing all the above measures, singletons are considered as additional one-member clusters.

11.6 A Case Study

In order to highlight the characteristics of each of the methods described above, we applied them to a yeast cell-cycle dataset containing the gene expression levels of yeast ORFs over 79 conditions. This dataset is available at http://cellcycle-www.stanford.edu.

The original dataset (Spellman et al. 1998) contains samples from yeast cultures synchronized by four independent methods: α factor arrest (samples taken every seven minutes for 119 minutes), arrest of a cdc15 temperature sensitive mutant (samples taken every 10 minutes for 290 minutes), arrest of a cdc28 temperature sensitive mutant (this part of the data is from Cho et al. 1998; samples taken every 10 minutes for 160 minutes), and elutriation (samples taken every 30 minutes for 6.5 hours). It also contains separate experiments in which G1 cyclin Cln3p or B-type cyclin Clb2p were induced.

Spellman et al. identified in this data eight hundred genes that are cell-cycle regulated (Spellman et al. 1998). The dataset that we used contains the expression levels of 698 out of those eight hundred genes, which have no missing entries, over the 72 conditions that cover the α factor, cdc28, cdc15, and elutriation experiments. (As in Tamayo et al. 1999, the 90-minutes datapoint was omitted from the cdc15 experiment.) Each row of the 698×72 matrix was normalized to have mean 0 and variance 1. (Note that by normalizing the variance different gene amplitudes are deemphasized and periodicity is more prominent.)

Based on the analysis conducted by Spellman et al., we expect to find in the data five main clusters: G1-peaking genes, S-peaking genes, G2-peaking genes, M-peaking genes, and M/G1-peaking genes. Each of these was shown to contain biologically meaningful subclusters.

The 698×72 dataset was clustered using five of the methods described above: K-means, SOM, CAST, hierarchical, and CLICK. The similarity measure used was the Pearson correlation coefficient. The authors of each of the programs were given the dataset and asked to provide a clustering solution. The identity of the dataset was not described and genes were permuted in an attempt to perform a "blind" test (although anyone familiar with the gene expression literature could have identified the nature of the data). The authors were told that the average homogeneity and average separation would be used to evaluate the quality of the solutions.

We present below the results for each of the methods. To allow the reader an impression of the results, we added for each of the clusterings (except the hierarchical one, which does not produce a hard partition of the elements) a reference figure prepared using MATLAB. This figure depicts the average pattern of the clusters along

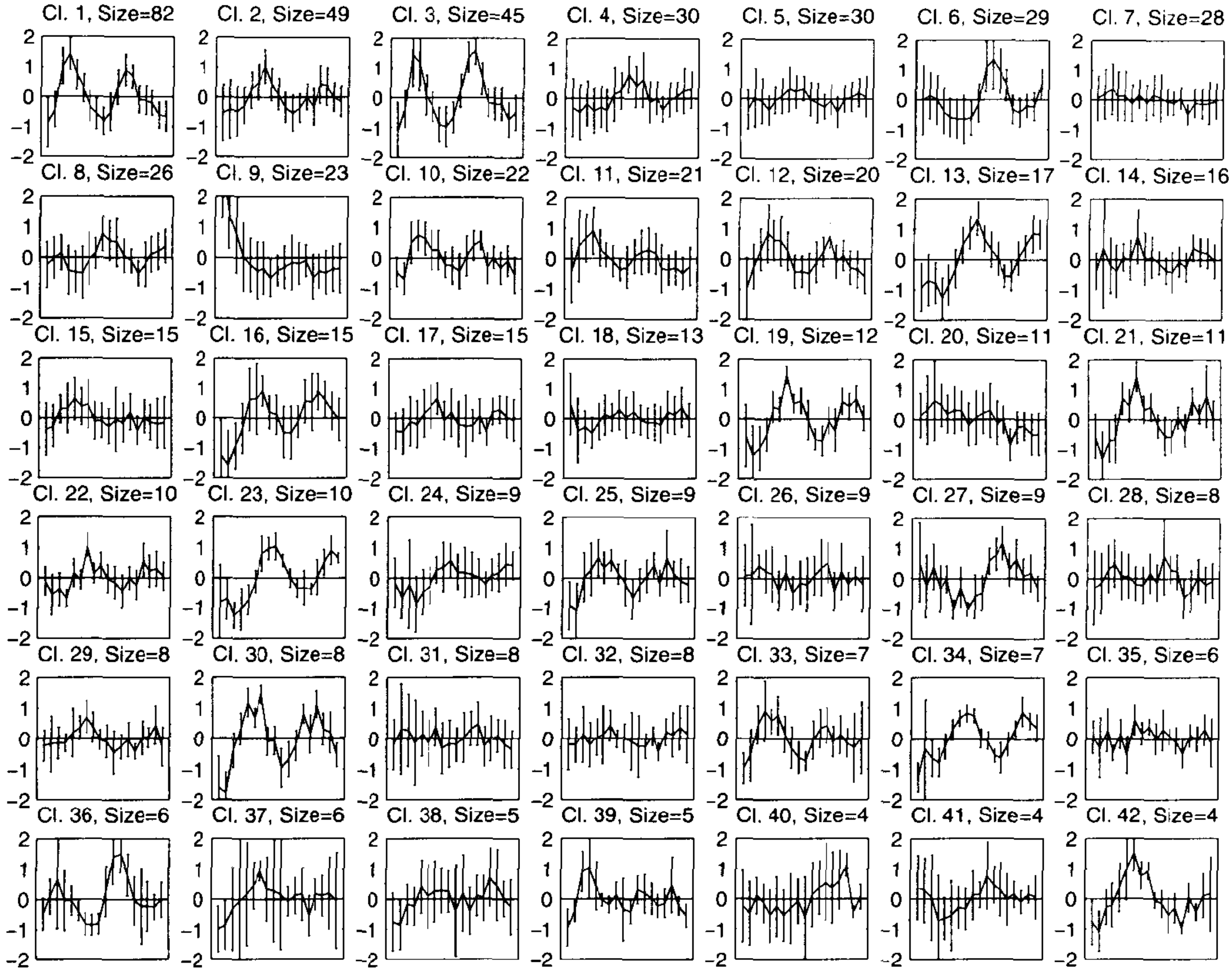


Figure 11.10
The clustering produced by the K-means algorithm of Herwig et al. x axis: time points 1–18 for the α factor experiment. y axis: normalized expression levels. The solid line in each subfigure plots the average pattern for that cluster. Error bars display the measured standard deviation. The cluster size is printed above each plot.

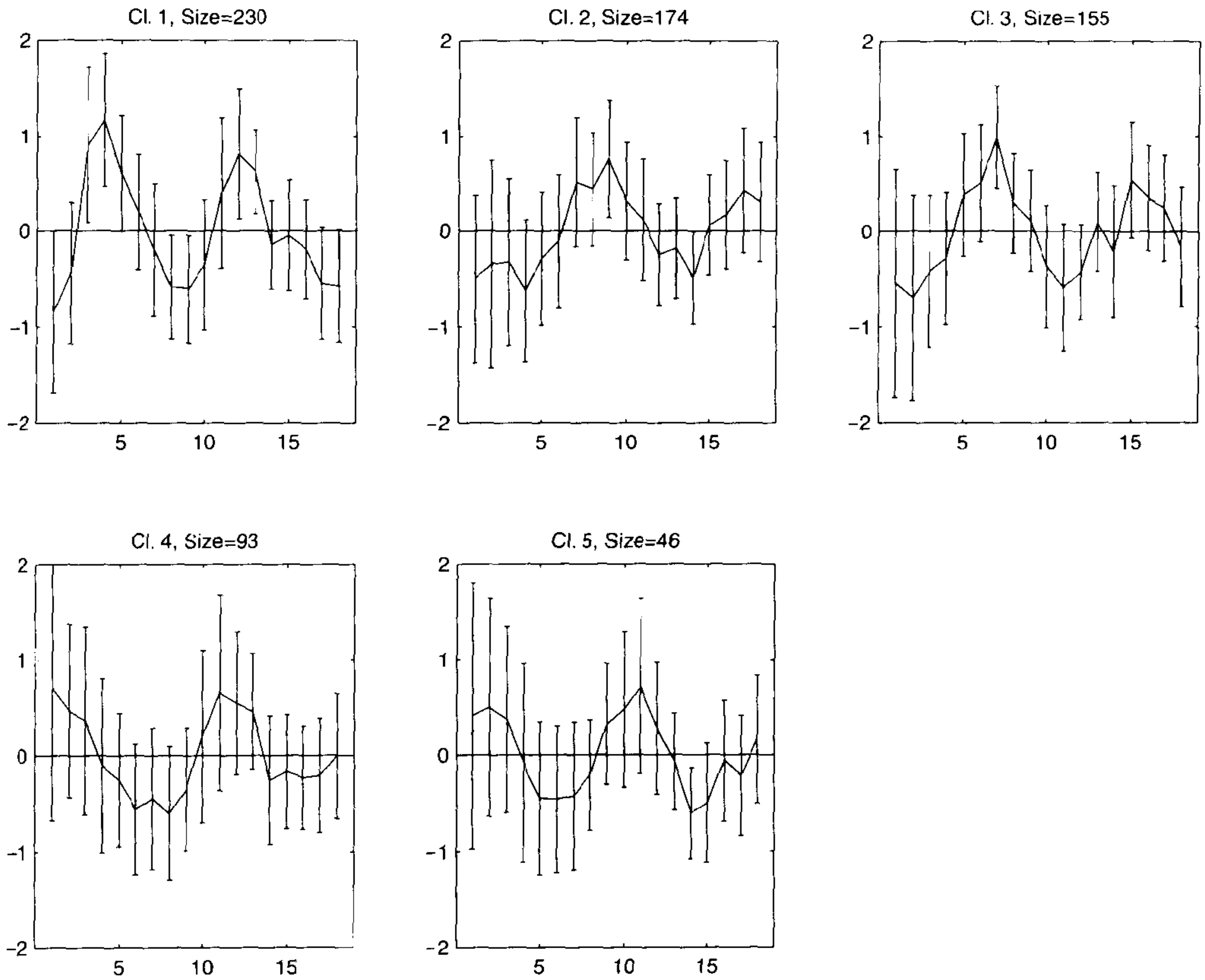


Figure 11.11
The clustering produced by the CAST algorithm of Ben-Dor et al.

with error-bars for the first 18 datapoints, which correspond to the α factor experiment. We have chosen not to show full expression patterns over all the 72 conditions, as these are much harder to interpret visually. Over the first 18 datapoints, one expects to view periodic behavior, with a distinct, typical pattern in each cluster. We also omitted from these figures small clusters with fewer than four members. As most programs output a variation of this figure, we have chosen to include the characteristic graphical output only for the programs Cluster and CAST.

The following table summarizes the solutions produced by each program (except for Cluster), and their homogeneity and separation parameters. The so-called "True"

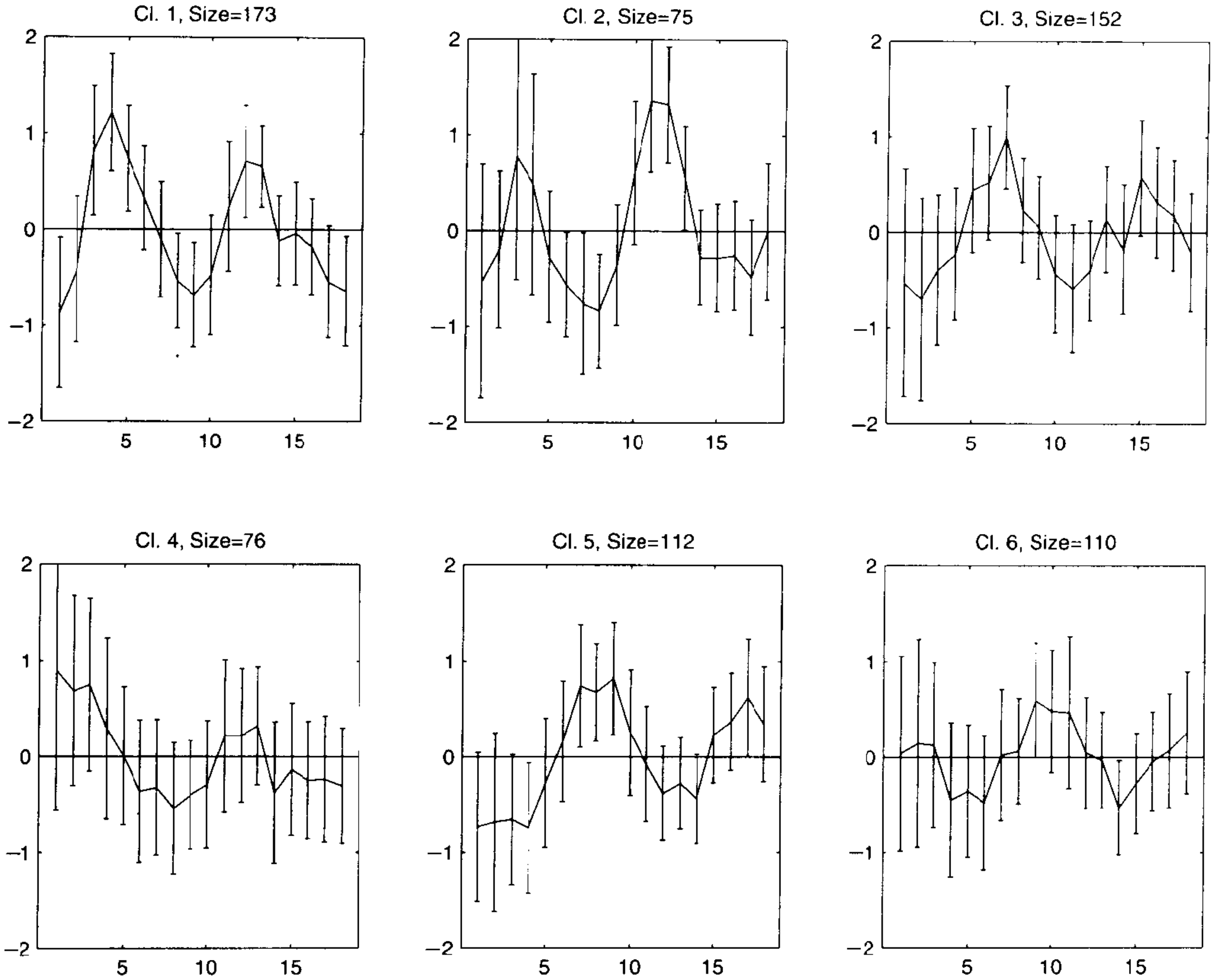


Figure 11.12
The clustering produced by the GeneCluster algorithm of Tamayo et al.

clustering, reported by Spellman et al. (1998) is that obtained manually by inspecting the expression patterns and comparing to the literature. The solution produced by CLICK contains 67 unclustered singletons.

The reference figures for each of the solutions are given in figures 11.10 to 11.14. The output of CAST is shown in figure 11.15. It depicts the similarity matrix before and after ordering its rows and columns based on the clustering. The output of Cluster is shown in figure 11.16. It includes a dendogram and a graphical representation of the ordered fingerprint matrix. (Experiments are also clustered and the solution is represented as a second dendogram on the same figure.) Figure 11.17 depicts the values of each solution on a plot of the homogeneity versus separation.

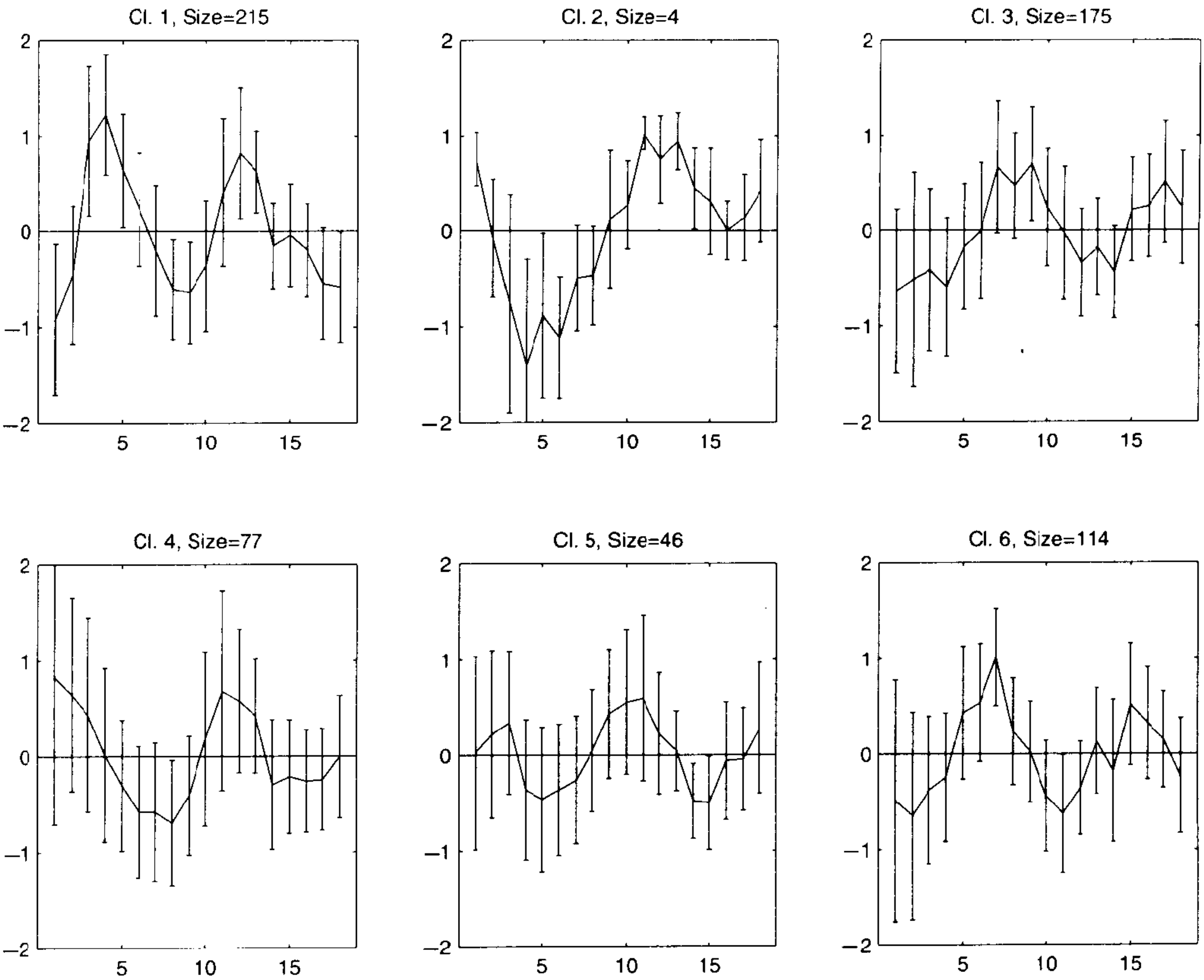


Figure 11.13
The clustering produced by the CLICK algorithm.

11.7 Concluding Remarks

Clustering remains, to certain extent, an art. There are no universal, agreed-upon criteria for evaluating solutions, and there is no ultimate algorithm. The clustering problem is so general, covering diverse disciplines and applications, that it is impossible to choose a single, "best" algorithm for solving the problem. This holds true even for the specific application of gene expression that we have addressed here. The eventual decision on what solution and what algorithm works best depends on the user and on the specific questions the clustering process is supposed to answer. Each of the algorithms that we have described has its strong points and its disadvantages. We shall address briefly below several key issues.

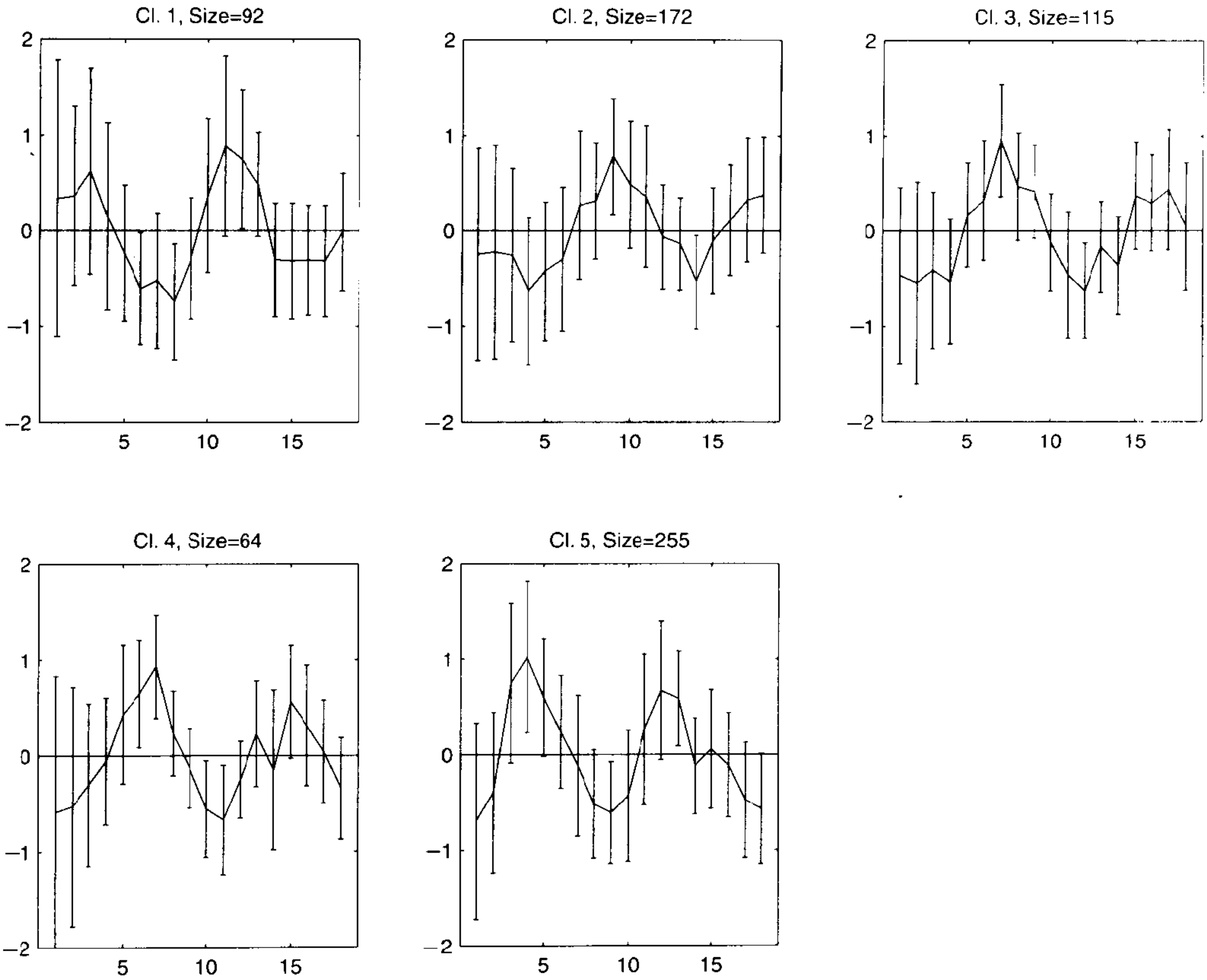


Figure 11.14
The "True" clustering of Spellman et al.

- *Choosing the clustering approach.* The hierarchical method is exceptional in our review, as it gives an overall view of the structure without an attempt to force a hard clustering. This can be viewed as an advantage or a disadvantage, depending on the experimental goals. The other methods aim to split the universe of elements into clusters, either by geometric approaches that move cluster centers (SOM, K-Means) or by using a graph theoretic approach. The latter may take a global view (CLICK) or single out one affinity-stable cluster at a time (CAST). As noted above, many other approaches were developed in other applications.
- *How should we evaluate solution quality?* We have described above several measures that evaluate solutions, both in the presence of a "correct" solution and in its

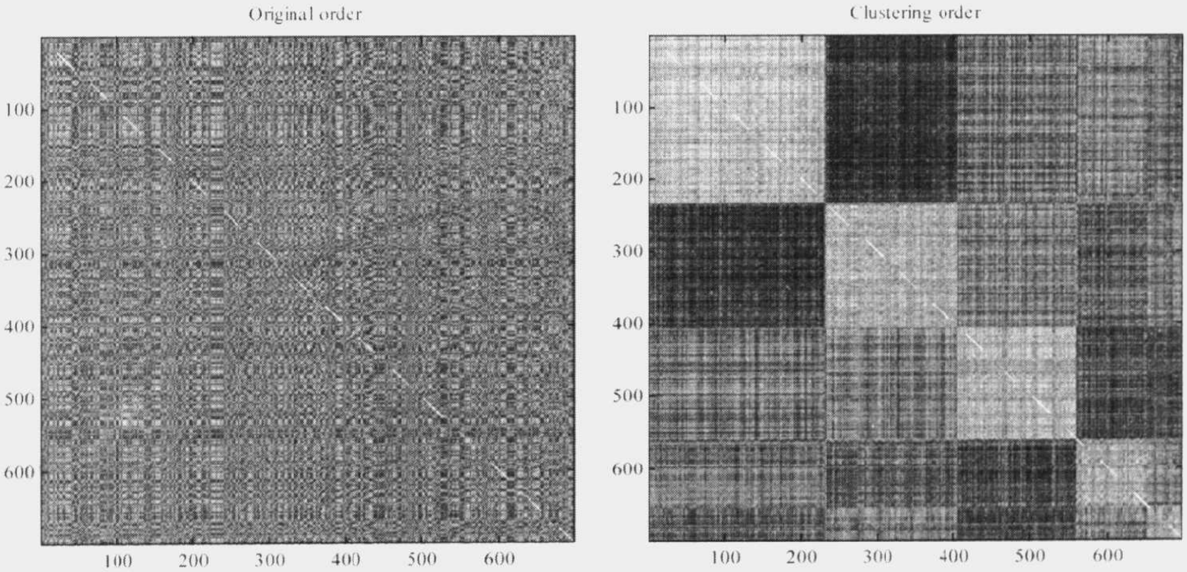


Figure 11.15
Representation of the solution produced by CAST. Left: the original similarity matrix. Right: the same matrix reordered according to the clustering. Grey level is inversely proportional to similarity. Genes belonging to the same cluster appear contiguously.

absence. The obvious advantage of having an objective function is the ability to compare solutions and measure progress in algorithm development. The caveat is that the measures may not reflect exactly the intuition that the biologist may have.

Even if one accepts the need of a numerical measure, the clustering literature is not in agreement on which measure to use, so we have presented two measures instead: an intra-cluster measure (homogeneity) and an inter-cluster measure (separation). The two are inherently conflicting, as an improvement in one will correspond to worsening of the other. One idea of overcoming this is by presenting a curve of homogeneity versus separation (A. Ben-Dor, private communication). Such a curve can naturally be obtained in CAST (by varying the single threshold parameter used) and can also be obtained by multiple runs of other algorithms. This curve can tell that one algorithm dominates another if it provides better homogeneity for all separation values, but typically each algorithm will dominate in a particular range. For another approach for comparing solutions across a range of parameters, see Yeung et al. (2000).

One way of getting around the "two objectives" problem is to fix the number of clusters. This is done by SOM and the classical K-means. When the number of clusters is known this is of course the way to go. When it is not known, what users often do is run such algorithms several times with several numbers of clusters (or grid topologies, in the case of SOM). However, this brings back the problem of evaluating and com-

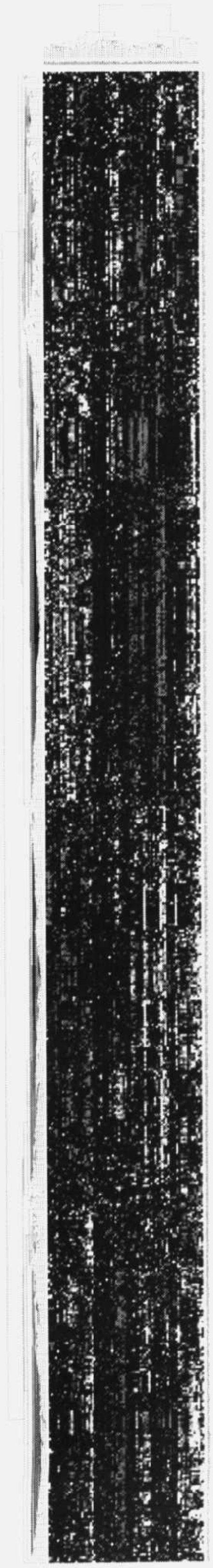


Figure 11.16
The output of Cluster. Actual output is in color, where red denotes increase vs. the reference level, and green denotes decrease. The gene clustering dendogram is on the top, and the experiment clustering dendogram is on the right.

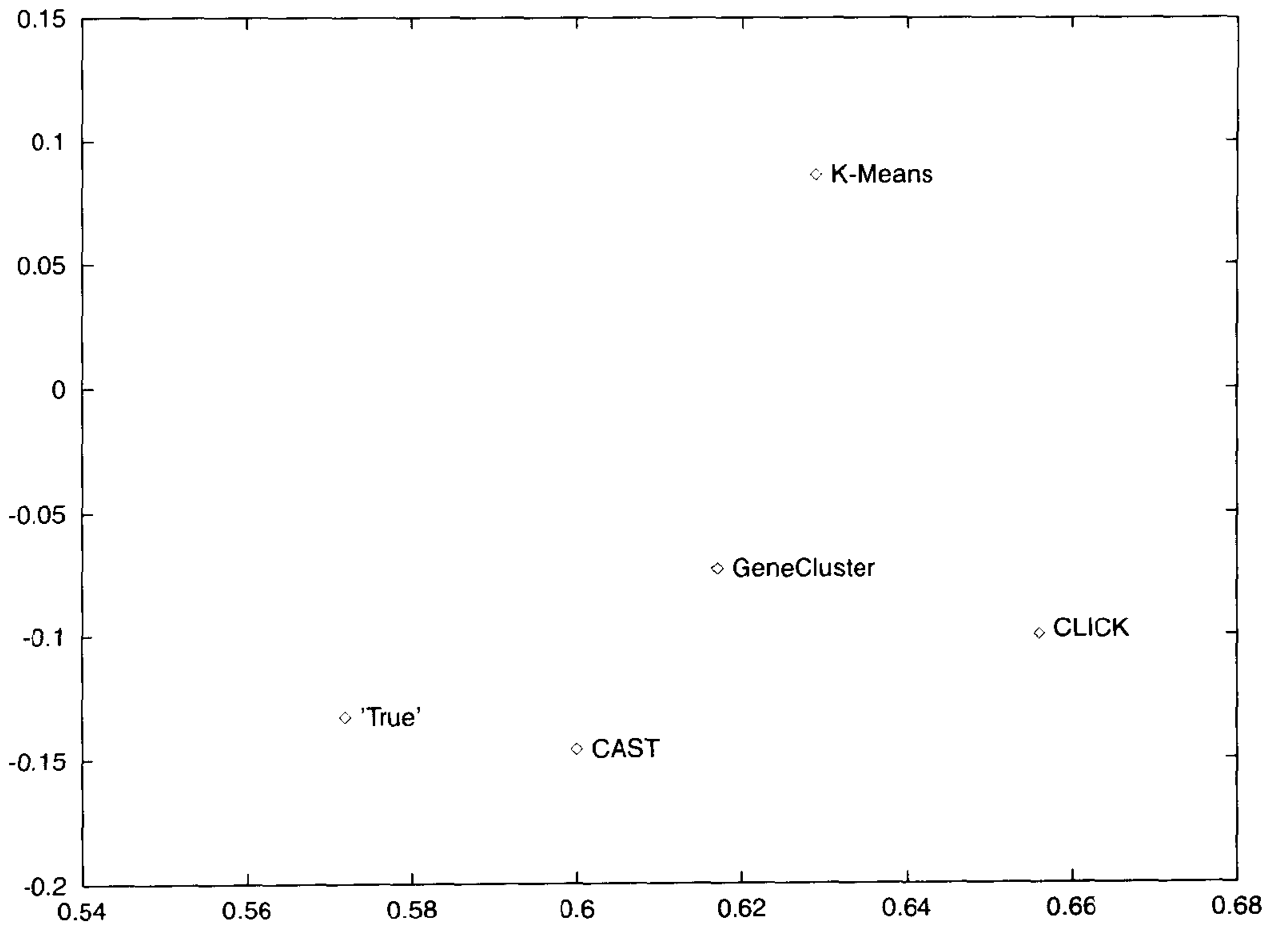


Figure 11.17
A comparison of homogeneity (x-axis) and separation (y-axis) values for all solutions. Recall that a solution improves if homogeneity increases or separation decreases.

paring solutions, so algorithms that seek a globally optimal solution seem preferable. Alternative methods of determining the number of clusters are given, for example, by Hartigan (1975) and Tibshirani et al. (2000).

• *Should we cluster all elements?* The SOM, K-means, and hierarchical algorithms require that the solution will constitute a partition of all the elements. Other algorithms, such as CLICK, allow some singletons to be left unclustered. By allowing singletons to be discarded, intra-cluster deviations can be reduced, perhaps at the expense of weaker separation. (Obviously, the number of discarded singletons must be kept to a small fraction of all elements, or else the solution would be meaningless.) In gene expression applications, one often does not seek an identification of *all* the genes involved, particularly as many genes have already been discarded in preprocessing steps, because of insignificant fingerprint variations. It is thus desirable to

Table 11.1
A summary of the clustering solutions and their figures of merit

Program	No. of Clusters	Homogeneity		Separation	
		H_{Ave}	H_{Min}	S_{Ave}	S_{Max}
K-Means	49	0.629	−0.339	0.086	0.911
CAST	5	0.6	0.037	−0.146	0.322
GeneCluster	6	0.617	0.067	−0.073	0.584
CLICK	6	0.656	0.097	−0.098	0.546
"True"	5	0.572	−0.322	−0.133	0.73

allow some room for discarding elements from a solution. It is not hard to add such flexibility into virtually all clustering algorithms that we have discussed.

• *Fingerprints versus similarity.* Some algorithms use only similarity values between elements, whereas others use the fingerprints themselves. Obviously, one loses some information by using the fingerprints to compute pairwise similarities only. One of the advantages of CLICK over HCS, for example, is by explicit use of the fingerprints for merging and adoption. Geometric algorithms like K-means and SOM use only fingerprints. Other algorithms like CAST may benefit from using such information more.

• *Visualization is crucial.* As the datasets and the solutions are very large, it is imperative to have tools to visualize summaries of the data and its solution from various viewpoints. The average patterns figures are useful to show trends, and SOM goes a step further by putting similar patterns in neighboring cells in a grid, generating a convenient "executive summary." The dendograms of Eisen et al. (1998) viewed together with the color-coded expression patterns of the genes are also very useful. Yet, devising additional novel, sophisticated (and ideally interactive) visualizations is an important challenge.

• *We need more testing data.* In order to improve the algorithms, we need more data. The best kind is actual gene expression data, along with a known clustering solution, so that it can be compared to the algorithmic solution. This is quite hard to obtain (except perhaps for oligofingerprint data) in the current status of biological knowledge. A second best is generating synthetic (simulated) datasets with known solutions, in which one can directly control individual parameters (cluster structure, errors, etc.). Some initial work has been done in this direction (Ben-Dor et al. 1999; Hartuv et al. 2000), but more work is needed in order to understand how to make the simulations realistic. Generating a publicly accessible benchmark of datasets—both synthetic and real—with known solutions, would be of great benefit to developing

better algorithms. In the absence of such resources, the available real data can be combined with evaluation methods as demonstrated here.

• *Clustering is only the first step.* In analyzing gene expression data, clustering is an essential initial step, but there is a lot more that can be done with the data. For example, one can use supervised learning techniques to cluster or classify the conditions. Such methods were recently shown to yield very good results in determining cancer types, with important potential applications to diagnostics (Golub et al. 1999; Alizadeh et al. 2000; Ben-Dor et al. 2000; Brown et al. 2000; Califano et al. 2000). Another useful idea is to cluster both the genes and the conditions, and to pinpoint subsets of the genes and the conditions ("biclustering") (Getz et al. 2000a; Cheng and Church 2000). Given the clusters, a variety of biological inference steps are possible. For example, identification of common control regions of upstream regions of genes from the same cluster (see chapter 10 of this book).

Acknowledgments

We thank Rani Elkon and Erez Hartuv for valuable help in preparing the manuscript. We also thank Ralf Herwig, Amir Ben-Dor, Michael Eisen, and Pablo Tamayo for providing us with the results of their algorithms. R. Shamir was supported in part by a grant from the Ministry of Science; Israel. R. Sharan was supported by an Eshkol fellowship from the Ministry of Science, Israel.

References

Alizadeh, A. A., Eisen, M. B., Davis, R. E., Ma, C., Lossos, I. S., Rosenwald, A., Boldrick, J. C., Sabet, H., Tran, T., Yu, X., Powell, J. I., Yang, L., Marti, G. E., Moore, T., Hudson, J., Lu, L., Lewis, D. B., Tibshirani, R., Sherlock, G., Chan, W. C., Greiner, T. C., Weisenburger, D. D., Armitage, J. O., Warnke, R., and Staudt, L. M. (2000). Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling. *Nature* 403(6769): 503–511.

Alon, U., Barkai, N., Notterman, D. A., Gish, G., Ybarra, S., Mack, D., and Levine, A. J. (1999). Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *PNAS* 96: 6745–6750.

Ball, G., and Hall, D. (1967). A clustering technique for summarizing multivariate data. *Behav. Sci.* 12(2): 153–155.

Ben-Dor, A., Shamir, R., and Yakhini, Z. (1999). Clustering gene expression patterns. *J. Comput. Biol.* 6(3/4): 281–297.

Ben-Dor, A., Bruhn, L., Friedman, N., Nachman, I., Schummer, M., and Yakhini, Z. (2000). Tissue classification with gene expression profiles. In *Proceedings of the Fourth Annual International Conference on Computational Moleculaer Biology (RECOMB 2000)*.

Benson, D. A., Boguski, M. S., Lipman, D. J., Ostell, J., Ouellette, B. F., Rapp, B. A., and Wheeler, D. L. (1999). Genbank. *Nucl. Acids Res.* 27(1): 12–17.

Brown, M. P., Grundy, W. N., Lin, D., Cristianini, N., Sugnet, C. W., Furey, T. S., Ares, M., and Haussler, D. (2000). Knowledge-based analysis of microarray gene expression data by using support vector machines. *Proc. Natl. Acad. Sci. USA* 97(1): 262–267.

Califano, A., Stolovitzky, G., and Tu, Y. (2000). Analysis of gene expression microarrays for phenotype classification. In *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology (ISMB)*, 75–85.

Cheng, Y., and Church, G. M. (2000). Biclustering of expression data. In *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology (ISMB)*, 93–103.

The chipping forecast. Special supplement to *Nature Genetics*, vol. 21, 1999.

Cho, R. J., Campbell, M. J., Winzeler, E. A., Steinmetz, L., Conway, A., Wodicka, L., Wolfsberg, T. G., Gabrielian, A. E., Landsman, D., Lockhart, D. J., and Davis, R. W. (1998). A genome-wide transcirptional analysis of the mitotic cell cycle. *Mol. Cell* 2: 65–73.

Cormack, R. M. (1971). A review of classification (with discussion). *J. Royal Statist. Soci. A* 134: 321–367.

Drmanac, S., and Drmanac, R. (1994). Processing of cDNA and genomic kilobase-size clones for massive screening mapping and sequencing by hybridization. *Biotechniques* 17: 328–336.

Drmanac, R., Lennon, G., Drmanac, S., Labat, I., Crkvenjakov, R., and Lehrach, H. (1991). Partial sequencing by oligohybridization: Concept and applications in genome analysis. In *Proceedings of the First International Conference on Electrophoresis Supercomputing and the Human Genome*, Cantor, C. and Lim H., eds. 60–75. Singapore: World Scientific.

Drmanac, S., Stavropoulos, N. A., Labat, I., Vonau, J., Hauser, B., Soares, M. B., and Drmanac, R. (1996). Gene-representing cDNA clusters defined by hybridization of 57419 clones from infant brain libraries with short oligonucleotide probes. *Genomics* 37: 29–40.

Eisen, M. B., and Brown, P. O. (1999). DNA arrays for analysis of gene expression. In *Methods in Enzymology, vol. 303*, 179–205.

Eisen, M. B., Spellman, P. T., Brown, P. O., and Botstein, D. (1998). Cluster analysis and display of genome-wide expression patterns. *PNAS* 95: 14863–14868.

Even, S. (1979). *Graph Algorithms*. Rockville, Md: Computer Science Press.

Everitt, B. (1993). *Cluster Analysis*. 3rd ed., London: Edward Arnold.

Fodor, S. P., Rava, R. P., Huang, X. C., Pease, A. C., Holmes, C. P., and Adams, C. L. (1993). Multiplexed biochemical assays with biological chips. *Nature* 364: 555–556.

Getz, G., Levine, E., and Domany, E. (2000). Coupled two-way clustering analysis of gene microarray data. *Proc. Natl. Acad. Sci. USA* 97(22): 12079–12084.

Getz, G., Levine, E., Domany, E., and Zhang, M. Q. (2000). Super-paramagnetic clustering of yeast gene expression profiles. *Physica* A279: 457.

Golub, T. R., Slonim, D. K., Tamayo, P., Hoard, C., Gaasenbeek, M., Mesirov, J. P., Coller, H., Loh, M. L., Downing, J. R., Caligiuri, M. A., Bloomfield, C. D., and Lander, E. S. (1999). Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science* 286: 531–537.

Golumbic, M. C. (1980). *Algorithmic Graph Theory and Perfect Graphs*. New York: Academic Press.

Hansen, P., and Jaumard, B. (1997). Cluster analysis and mathematical programming. *Math. Program.* 79: 191–215.

Harrington, C. A., Rosenow, C., and Retief, J. (2000). Monitoring gene expression using DNA microarrays. *Curr. Opin. Microbiol.* 3(3): 285–291.

Hartigan, J. A. (1975). *Clustering Algorithms*. New York: John Wiley and Sons.

Hartuv, E., and Shamir, R. (1999). A clustering algorithm based no graph connectivity. Tel Aviv University, Dept. of Computer Science, Technical report.

Hartuv, E., Schmitt, A., Lange, J., Meier-Ewert, S., Lehrach, H., and Shamir, R. (2000). An algorithm for clustering cDNA fingerprints. *Genomics* 66(3): 249–256.

Herwig, R., Poustka, A. J., Meuller, C., Lehrach, H., and O'Brien, J. (1999). Large-scale clustering of cDNA-fingerprinting data. *Genome Res.* 9(11): 1093–1105.

Heyer, L. J., Kruglyak, S., and Yooseph, S. (1999). Exploring expression data: Identification and analysis of coexpressed genes. *Genome Res.* 9(11): 1106–1115.

Kerr, M. K., Martin, M., and Churchill, G. A. (2000). Analysis of variance for gene expression microarray data. Technical report, Jackson Laboratory.

Kohonen, T. (1997). *Self-Organizing Maps.* Berlin: Springer.

Lance, G. N., and Williams, W. T. (1967). A general theory of classification sorting strategies. 1. Hierarchical systems. *Comput. J.* 9: 373–380.

Lennon, G. S., and Lehrach, H. (1991). Hybridization analysis of arrayed cDNA libraries. *Trends Genet.* 7: 60–75.

Lipshutz, R. J., Fodor, S. P. A., Gingeras, T. R., and Lockhart, D. J. (2000). High density synthetic oligonucleotide arrays. *Nature Genet. Suppl.* 21: 20–24.

Lockhart, D. J., and Winzeler, E. A. (2000). Genomics, gene expression and DNA arrays. *Nature* 405(6788): 827–836.

MacQueen, J. (1965). Some methods for classification and analysis of multivariate observations. In *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, 281–297.

Marshall, A., and Hodgson, J. (1998). DNA chips: An array of possibilities. *Nat. Biotechnol.* 16: 27–31.

Meier-Ewert, S., Mott, R., and Lehrach, H. (1995). Gene identification by oligonucleotide fingerprinting a pilot study. MPI, technical report.

Milosavljevic, A., Strezoska, Z., Zeremski, M., Grujic, D., Paunesku, T., and Crkvenjakov, R. (1995). Clone clustering by hybridization. *Genomics* 27: 83–89.

Mirkin, B. (1996). *Mathematical Classification and Clustering.* Boston: Kluwer.

Poustka, A. J., Herwig, R., Krause, A., Hennig, S., Meier-Ewert, S., and Lehrach, H. (1999). Toward the gene catalogue of sea urchin development: The construction and analysis of an unfertilized egg cDNA library highly normalized by oligonucleotide fingerprinting. *Genomics* 59: 122–133.

Ramsay, G. (1998). DNA chips: State-of-the art. *Nat. Biotechnol.* 16: 40–44.

Schena, M. (1996). Genome analysis with gene expression microarrays. *Bioessays* 18: 427–431.

Schena, M., Shalon, D., Heller, R., Chai, A., Brown, P. O., and Davis, R. W. (1996). Parallel human genome analysis: Microarray-based expression monitoring of 1000 genes. *Proc. Natl. Acad. Sci. USA* 93: 10614–10619.

Schuler, G. D. (1997). Pieces of the puzzle: Expressed sequence tags and the catalog of human genes. *J. Mol. Med.* 75(10): 694–698.

Sharan, R., and Shamir, R. (2000). CLICK: A clustering algorithm for gene expression analysis. In *Currents in Compuational Molecular Biology*, Miyano, S., Shamir, R., and Takagi, T., eds. 6–7. Tokyo: Universal Academy Press.

Sharan, R., and Shamir, R. (2000). CLICK: A clustering algorithm with applications to gene expression analysis. In *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology (ISMB)*, 307–316.

Sokal, R. R. (1977). Clustering and classification: Background and current directions. In *Classification and Clustering*, Van Ryzin, J., ed. 1–15. New York: Academic Press.

Spellman, P. T., Sherlock, G., Zhang, M. Q., Lyer, V. R., Anders, K., Eisen, M. B., Brown, P. O., Botstein, D., and Futcher, B. (1998). Comprehensive identification of cell cycle-regulated genes of the yeast Saccharomyces cerevisiae by microarray hybridization. *Mol. Biol. Cell* 9: 3273–3297.

Tamayo, P., Slonim, D., Mesirov, J., Zhu, Q., Kitareewan, S., Dmitrovsky, E., Lander, E. S., and Golub, T. R. (1999). Interpreting patterns of gene expression with self-organizing maps: Methods and application to hematopoietic differentiation. *PNAS* 96: 2907–2912.

Tibshirani, R., Walther, G., and Hastie, T. (2000). Estimating the number of clusters in a dataset via the gap statistics. Stanford University, technical report.

Toronen, P., Kolehmainen, M., Wong, G., and Castren, E. (1999). Analysis of gene expression data using self-organizing maps. *FEBS Letters* 451: 142–146.

Drmanac, R., Drmanac, S., Labat, I., Crkvenjakov, R., Vicentic, A., and Gemmell, A. (1992). Sequencing by hybridization: Towards an automated sequencing of one milion M13 clones arrayed on membranes. *Electrophoresis* 13: 566–573.

Yeung, K. Y., Haynor, D. R., and Ruzzo, W. L. (2000). Validating clustering for gene expression data. University of Washington, technical Report UW-CSE-00-01-01.

12 KEGG for Computational Genomics

Minoru Kanehisa and Susumu Goto

12.1 Introduction

Once a complete genome sequence is known, it should in principle be possible to identify all the genes and uncover all their functions by computational methods. In reality, this is not possible. Is it because the current computational methods are imperfect or because the information in the genome is insufficient? Whichever to believe, anyone would agree that additional data and knowledge will help to interpret the complete genome sequence information. Post-genomics is an emerging field for developing new experimental and computational technologies, such as DNA chips and protein chips, for generating and analyzing different types of systematic data, such as gene expressions and polymorphisms, and for expanding our biological knowledge based on the genomic information. Here again, as figure 12.1 illustrates, differences arise depending on the views or directions taken for post-genome analyses.

In the traditional view of molecular biology, after the genome is the transcriptome, and then the proteome. The transcriptome represents a whole set of mRNAs expressed in the cell of a given tissue under a given condition. The proteome usually represents a whole set of proteins expressed in the cell and how they interact with each other, but it may also mean structural genomics to systematically determine a catalog of protein 3D structures. Computational molecular biology has been the discipline of choice to analyze sequence and 3D structural information of DNAs, RNAs, and proteins in order to understand molecular functions. However, the analysis of individual molecules would never be sufficient for understanding higher order functions of cells and organisms, represented by another axis in figure 12.1. Furthermore, although the biological macromolecules of DNAs, RNAs, and proteins may play major roles, there are other substances that together make up the entire chemical complement of the cell. The third axis emphasizes the roles of chemical compounds and metal ions in biological functions.

The two additional axes in figure 12.1 represent an extension of the traditional molecular biology (Kanehisa 2000a); they are in fact the conceptual basis of KEGG, the Kyoto Encyclopedia of Genes and Genomes (Kanehisa 1997a). In our view, the genome is simply an information storage of how to make individual molecular building blocks of life. The genome does not contain much information about the wiring of building blocks—for example, how they interact to make up a cell or to exert cellular functions. The wiring information is likely to be distributed in the cell and more dynamic in nature. Although the molecular wiring diagram of the cell may

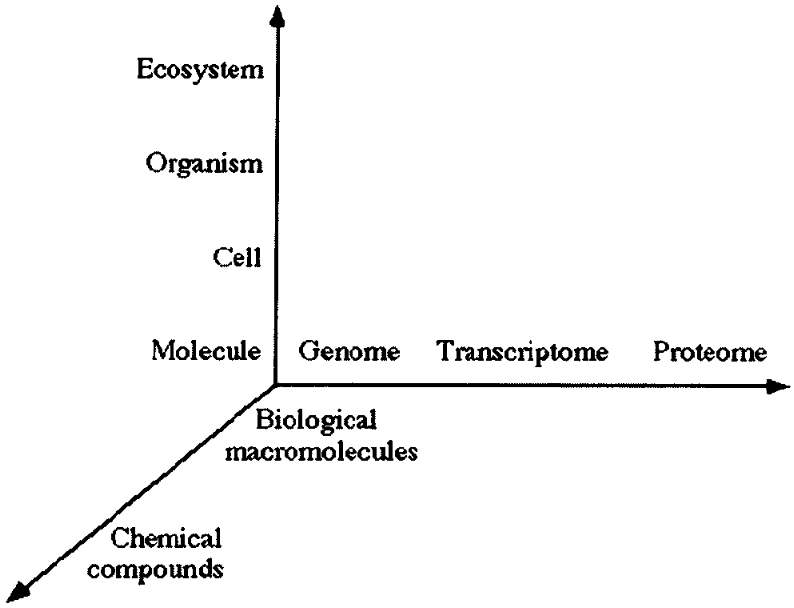


Figure 12.1
Post-genomics in three directions.

not be computable from the information in the genome alone, it may still be predictable, at least to some extent, if we have sufficient knowledge of actual wiring in living cells and if empirical relations to genomes can be found. Thus, we have been computerizing current knowledge on molecular pathways and complexes in the PATHWAY database, and analyzing possible relations to the gene catalogs of all the completely sequenced genomes and some partial genomes that are stored in the GENES database in KEGG. We have also been collecting information about chemical compounds and chemical reactions in the LIGAND database (Goto et al. 1998). Such information is essential for understanding the dynamic interactions of the cell with its environment.

In traditional computational molecular biology, the data objects to be analyzed consist of elements that are abstracted to symbols at the atomic level, such as C for carbon in the protein 3D structure, or to other symbols at the molecular level, such as C for cysteine in the amino acid sequence. In computational genomics as we define here, the data objects to be analyzed are the genome, which is a sequence of genes, the pathway that is a network of interacting proteins, and other types of relations among genes or gene products. Thus, symbols are used for an abstract representation of data elements at a higher level, such as polA for a gene in the genome and Ras for a protein in the pathway. KEGG is a computational resource for analyzing networks

Table 12.1
Examples of biological complex systems at different levels

Complex system	Node	Edge (interaction)
Protein	Atom	Atomic interaction
Cell	Molecule	Molecular interaction
Brain	Cell	Cellular interaction
Ecosystem	Organism	Organism interaction
Civilization	Human	Human interaction

of such higher level symbols. It is highly integrated with the existing molecular biology resources for analyzing sequences of molecular symbols and networks of atomic symbols.

12.2 KEGG Ontology

12.2.1 Complex Systems

Life is a manifestation of biological complex systems at different levels, as exemplified in table 12.1. A complex system consists of nodes and edges, namely, building blocks and their interactions, and it is interacting with the environment. The protein is a complex system consisting of atoms and atomic interactions. Under the physiological environment, the protein assumes the native 3D structure, which makes it possible to perform a specific biological function. When the environment is perturbed, the native structure is disrupted and the protein loses its function. The structural change occurs in a narrow range of environmental conditions, which is like a phase transition in physical phenomena and which represents a systemic behavior of the complex system.

At a higher level of abstraction, the cell may be viewed as a complex system consisting of molecules and molecular interactions (table 12.1). When there is a proper network of molecules, such as a series of enzymes catalyzing successive reaction steps in a metabolic pathway or a set of proteins that forms a signal transduction pathway, then the cell is able to perform its specific function, such as biosynthesis of amino acids or response to environmental stresses. Thus, the specific network of molecules in the cell can be related to a higher order cellular function, which is like relating the specific 3D structure of the protein to its molecular function. When the cell is perturbed, for example, by a foreign substance in the environment or a mutation in the genome, a dynamic change may be observed in the global network of interacting molecules. Such a systemic response to a perturbation is again a common feature of complex systems.

Table 12.2
Graph representation of KEGG data objects

Graph (data object)	Node	Edge (interaction or relation)
Genome	Gene	Ajacency
Transcriptome	Gene	Expression similarity
Proteome	Protein	Direct interaction
Protein universe	Protein	Sequence similarity or 3D structural similarity
Gene universe	Gene	Orthology, paralogy, or xenology
Complex	Gene product (protein or RNA)	Direct interaction
Pathway	Gene product or complex	Generalized protein-protein interaction (direct interaction, gene expression relation, or enzyme-enzyme relation)

Network ::= Pathway | Complex

12.2.2 Graph Representation

Although there are other systemic phenomena at still higher levels (table 12.1), we focus our analysis on the level of molecular interactions because this is the level where the information in the genome can be directly correlated. We also extend the concept of edges to other types of relations. Thus, many data objects in KEGG are represented by a graph, which is a set of nodes and edges, as summarized in table 12.2. The genome is a graph consisting of one-dimensionally connected nodes (genes). The transcriptome generated by systematic gene expression profile analyses can be interpreted as a graph of expression similarity from which clusters of coregulated genes can be identified. The proteome obtained by yeast two hybrid system experiments or mass spectroscopy experiments suggest a graph of possible protein-protein interactions in complexes and pathways. In addition to experimental data on genomes, transcriptomes, and proteomes, the result of computational analyses can also be represented by a graph, such as the protein universe viewed as a hierarchy of structurally similar proteins and the gene universe representing evolutionary relations of genes and organisms.

One of the major objectives of KEGG is to computerize data and knowledge on molecular pathways and complexes that are involved in various cellular processes. Thus, KEGG contains a unique data object termed the generalized protein-protein interaction network, or simply the network, which is an abstract network of gene products (Kanehisa 2000a, b). Although there may be different ways of representing a network of interacting molecules in the cell, the representation in KEGG focuses on proteins and RNAs that are directly linked to genes in the genome. As shown in figure 12.2, the generalized protein-protein interaction includes: (a) a direct interaction such as binding, modification, or cleavage; (b) an indirect interaction involving

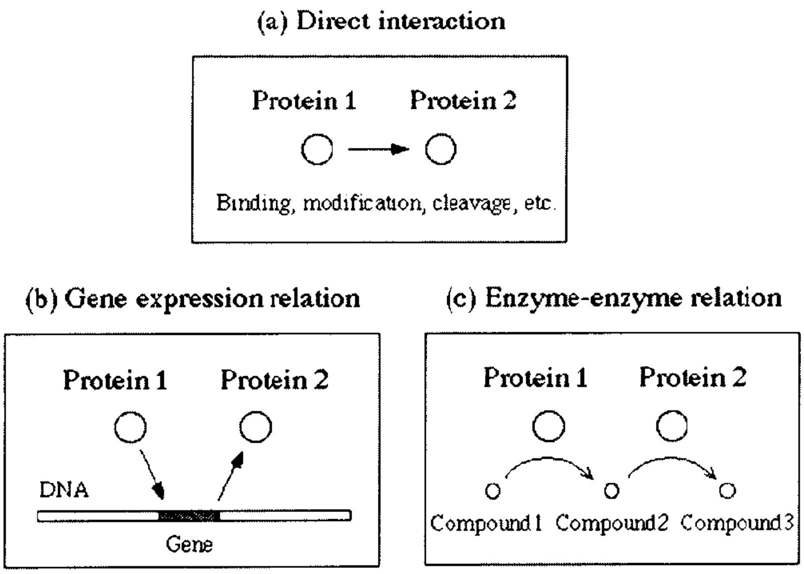


Figure 12.2
The network of interacting molecules in the cell, such as a pathway or a complex, is represented in KEGG by the three types of generalized protein-protein interactions.

gene expression, namely, the relation between a transcription factor and a target gene product; and (c) another indirect interaction representing the relation of two enzymes that catalyze two successive reaction steps. It must be noted that DNAs and chemical compounds are not considered as the nodes of the network, but rather they are part of the edges. Of course, details of protein-DNA interactions in gene expressions and protein-ligand interactions in enzymatic reactions are useful information in actual data analysis. It must also be noted that the term protein used here actually includes an RNA or a complex of proteins and/or RNAs.

12.2.3 Functional Hierarchy

The graph representation shown in table 12.2 is the most basic content of the KEGG ontology, which is a formal specification of entities and their relations in KEGG. In addition, the network (pathway or complex) is hierarchically structured, as shown in figure 12.3. The three categories in the top hierarchy, metabolism, genetic information processing, and environmental information processing, are the three essential aspects of life in any organism; the fourth category of cellular processes contains divergent aspects of cellular functions in various organisms. As of October 2000, the KEGG network hierarchy has been further subdivided into 21 subcategories (figure 12.3) and over 120 sub-subcategories (not shown).

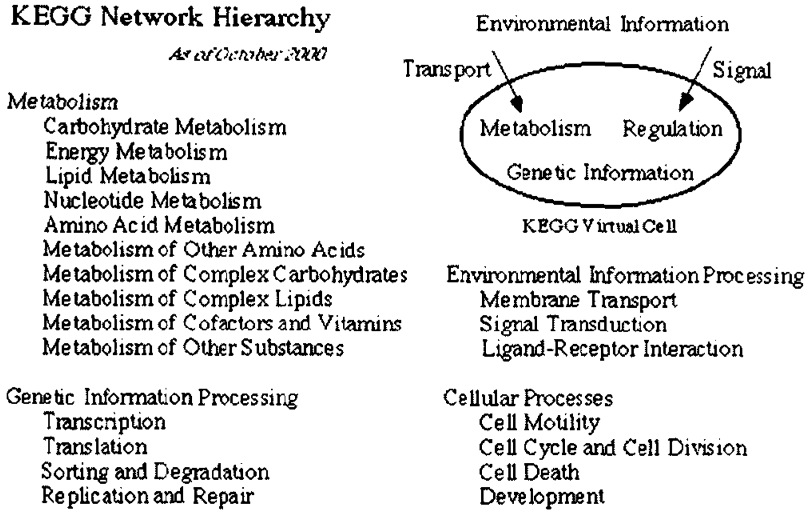


Figure 12.3
The functional hierarchy of the KEGG network data. The top two levels are shown here.

Once the complete genome sequence is determined, it has become customary to present a hierarchical classification of gene functions. In contrast to various existing classification schemes (Riley 1993; Ashburner et al. 2000), the KEGG functional hierarchy is assigned to the network, rather than to individual genes, because any higher order function involving a cell or an organism is an attribute of the network. KEGG does provide a hierarchical classification of genes for each genome, which is automatically generated in the process of matching genes in the genome and gene products in the network (see below).

12.3 Suite of KEGG Databases

12.3.1 PATHWAY Database

The KEGG ontology summarized above is implemented in the databases shown in table 12.3, which are all available at the GenomeNet (http://www.genome.ad.jp/). The main database PATHWAY is a collection of known pathways (and complexes) that are involved in various cellular processes. Mostly from the literature, a pathway is drawn manually as a graphical diagram based on the concept of the generalized protein-protein interaction network. This is, what is called a reference pathway from which a number of organism-specific pathways are computationally generated by matching against individual genes in the genome. At the moment, there are about two hundred reference pathways; each pathway contains, on the average, about 30 proteins.

Table 12.3
KEGG databases

Data object	Database	Data type	Content
Network	PATHWAY	Graph	Generalized protein-protein interaction networks for various cellular processes
Genome	GENES	Node	Gene catalogs for completely sequenced genomes and some partial genomes
	GENOME	Graph	Genome maps and information about organisms
Transcriptome	EXPRESSION	Graph	Microarray gene expression profiles
Proteome	BRITE	Graph	Protein-protein interactions and relations
Environment	LIGAND	Edge	Chemical compounds and chemical reactions

The KEGG pathways are divided into metabolic pathways and regulatory pathways, which correspond to metabolism and the rest, respectively, in the functional hierarchy shown in figure 12.3. A metabolic pathway involving enzymatic reactions on chemical substances consists of enzyme-enzyme relations, whereas a regulatory pathway involving macromolecular reactions and interactions mostly consists of direct interactions and gene expression relations. For computational purposes of using KEGG metabolic pathways, an auxiliary file is provided containing an entire list of enzyme-enzyme relations. Such binary relation files are not yet available for regulatory pathways.

12.3.2 GENES Database

The GENES database provides the gene catalog information for all the completely sequenced genomes and some partial genomes, including human and mouse. An entry in GENES contains sequence information and functional annotation, together with links to the PATHWAY and GENOME databases in KEGG and to other outside databases. When the complete genome sequence is publicly made available in GenBank, it is incorporated in the GENES database within a few days, and the assignment of EC numbers and ortholog identifiers is performed within a few weeks. Here the ortholog identifier is an extension of the EC numbering system for enzymes (Kanehisa and Goto 2000). It is applicable to all proteins and RNAs, and it can distinguish subunits or genes with the same EC number.

Once the assignment of ortholog identifiers is manually performed, organism-specific pathways are automatically generated by matching genes in the genome and gene products in the pathway. They are represented by the coloring of gene product nodes (boxes) in the reference pathways. This is possible because each node in the

pathway is associated either with an ortholog identifier or with the combination of an organism name and a gene name. The matching process also generates a gene catalog for each organism, which is a hierarchical classification of genes according to the functional hierarchy of KEGG pathways (figure 12.3). The gene catalog is manipulated by what is called the hierarchical text browser. For most genomes, the hierarchical classification of gene functions provided by the original authors is also made compatible with the hierarchical text browser.

After the initial annotation of ortholog identifiers, efforts are continuously made to standardize the terminology across organisms and to provide the most up-to-date information according to new experimental evidence reported in the literature, convincing results of our pathway analysis, and functional annotations of the SWISS-PROT and other databases. The gene annotations are maintained by the Web-based KEGG annotation tool, which is linked to a relational database and which is integrated with GFIT (Bono et al. 1998) and other computational tools.

12.3.3 GENOME Database

The GENOME database is a collection of genome maps containing information about chromosomal locations of genes for completely sequenced genomes. The genome map is manipulated by the Java-based genome map browser. There are again two versions of genome maps, original and KEGG, corresponding to the two versions of the gene catalogs. They differ in the coloring of genes that represent functional hierarchy and also the links made to individual gene entries. The GENOME database is associated with the taxonomy and text information about each organism.

12.3.4 Ortholog Group Table

The KEGG ortholog group table is a condensation of the results obtained by the integrated analysis of the PATHWAY, GENES, and GENOME databases. In contrast to efforts such as the COG database (Tatusov et al. 1997), which attempts to classify all genes into clusters of orthologous genes, the concept of the ortholog group is applied here to sets of functionally correlated genes, such as orthologous operons, rather than to individual genes. The ortholog group table was first constructed for each conserved portion of the metabolic pathway that was identified as a correlated cluster (see below) of genes in the genome and gene products in the pathway; for example, a set of genes in an operon responsible for a biosynthetic pathway (Ogata et al. 2000). The collection of ortholog group tables was then expanded to other pathways and molecular complexes by examining correlated clusters of genes in multiple genomes (Fujibuchi et al. 2000), also based on knowledge in the literature.

The ortholog group table is an HTML table with an embedded manipulation program for row-wise and column-wise operations. Each row indicates whether genes are present or not for a given organism and also whether there are adjacent genes in the genome, possibly forming operons, by coloring. Each column contains a set of orthologous genes based not simply on sequence similarity but also on the positional correlation of genes and the completeness of the pathway. Thus, the compilation of ortholog group tables has been extremely useful in identifing unannotated or misannotated genes in the original databases. The table can also be viewed as a multiple alignment of organism-specific pathways, indicating a pathway motif or a functional unit of the cellular processes.

12.3.5 EXPRESSION Database

The EXPRESSION database is a new addition to the KEGG system. It is being developed for our ongoing project to analyze microarray gene expression profiles in *Saccharomyces cerevisiae*, *Synechocystis* PCC6803, *Bacillus subtilis*, and *Escherichia coli*. An entry in EXPRESSION corresponds to a piece of hybridization data, which can be viewed and analyzed in combination with the PATHWAY and GENOME information by the Java-based expression browser.

12.3.6 BRITE Database

BRITE (Biomolecular Relations in Information Transmission and Expression) is a database of binary relations between proteins or other biological molecules. The concept of binary relations, which is equivalent to the concept of edges (tables 12.1 and 12.2), has also been used in the DBGET/LinkDB system (Kanehisa 1997b; Fujibuchi et al. 1998) to compute indirect (deduced) links between databases. BRITE is still at an early stage of development, but it aims at enhancing such deductive database capabilities for biological relations of genes and gene products. In view of the developments in experimental technologies for protein-protein interactions, there will be a huge amount of biological binary relation data that will be part of the BRITE database.

12.3.7 LIGAND Database

The role of the LIGAND database (Goto et al. 1998) in the KEGG system has been to provide detailed molecular information about one type of the generalized protein-protein interaction, namely, the enzyme-enzyme relation. LIGAND is a composite database of ENZYME and COMPOUND. The ENZYME section stores the information about enzymatic reactions and enzyme molecules according to the up-to-date classification of the EC numbers, whereas the COMPOUND section is a collection

of about six thousand chemical compounds, most of which are metabolites in the metabolic pathways. The ENZYME and COMPOUND entries are linked from the KEGG reference pathways for metabolism, thus providing molecular details of network information.

The future role of LIGAND is to integrate the information about the environment of the network. We will organize data and knowledge of chemical compounds and chemical reactions that affect living cells and organisms, including drugs, environmental compounds, and their metabolisms, in the COMPOUND section and the third REACTION section of the LIGAND database.

12.3.8 Hierarchical Classifications

The data objects shown in table 12.2 are collected in the KEGG databases shown in table 12.3, except for the protein universe and the gene universe. These data objects have been derived from the sequence and 3D structure databases; KEGG just makes use of the existing compilations. For example, the hierarchy of protein folds and sequence similarities in the SCOP database (Murzin et al. 1995) can be used to analyze pathway information by the hierarchical text browser in KEGG.

In addition to hierarchically classified gene catalogs and protein catalogs, other types of classifications, such as diseases and cell types, are being integrated in KEGG in order to make links between genotypes and phenotypes.

12.4 Graph Comparison and Network Prediction

12.4.1 Graph Comparison to Detect Correlated Clusters

Sequence comparison has been the most powerful method to identify molecular functions of proteins and nucleic acids. At the network level of interacting molecules, because all the data objects are represented by graphs (table 12.2), the graph comparison is bound to become the most powerful method to understand higher order functions. We have developed a heuristic graph comparison algorithm to detect certain graph similarities called correlated clusters (Ogata et al. 2000). In contrast to the standard notion of graph similarity, or graph isomorphism, this algorithm detects loose similarities that are biologically more relevant by allowing gaps and mismatches.

A cluster is a set of nodes that are closely positioned in a graph. A correlated cluster is a set of clusters in two or more graphs whose nodes are correlated by certain relations. For example, when comparing the genome graph and the pathway graph, the correspondence of nodes is given by the relation of genes to gene products. The resulting correlated cluster will be a set of genes that are adjacent in the genome and whose protein products are functioning at close positions in the pathway, such as a

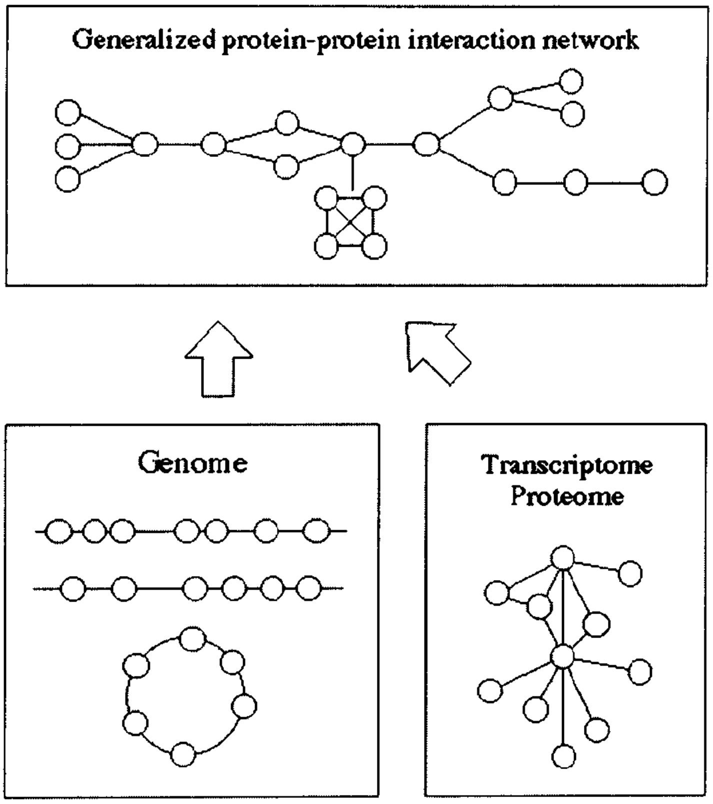


Figure 12.4
The network prediction is formulated as a conversion of the genome graph with genes as nodes to the network graph with gene products as nodes. The prediction is based on the reference knowledge of similar networks as well as sets of binary relations in transcriptomes and proteomes.

specific pathway coded by an operon. When comparing the genome graph of one organism to the genome graph of another organism, the correspondence may be given by the amino acid sequence similarity. The resulting cluster may then be a conserved operon consisting of similar genes. Note that the order of individual genes in each operon does not have to be conserved, which is the essence of loose similarity.

12.4.2 Network Prediction from Genomic Information

The network prediction in KEGG is to compute the generalized protein-protein interaction network, or the network of gene products, from the catalog of genes in the genome, as illustrated in figure 12.4. The prediction is based on the reference knowledge of real networks in the PATHWAY database and additional information of transcriptomes and proteomes in the EXPRESSION and BRITE databases. The problem can be viewed as a conversion of the genome graph to the network graph by integrating additional graphs of transcriptomes, proteomes, and similar networks.

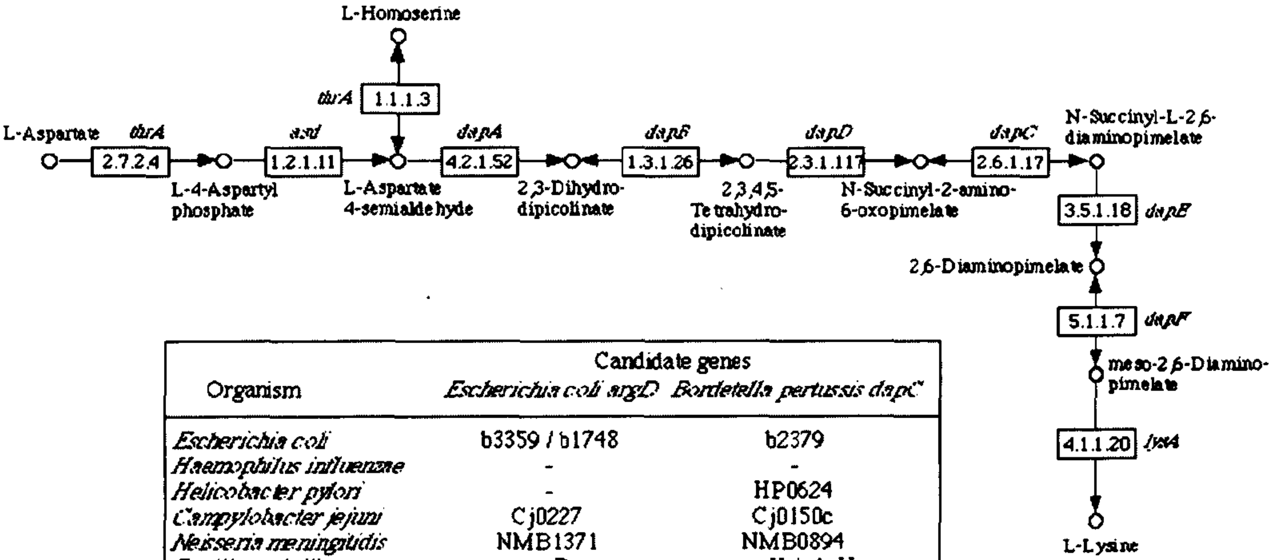
Thus, the graph comparison is an essential feature to integrate different information represented by different types of graphs.

Although we do not yet have a fully automated method, the current KEGG databases and computational tools can be utilized for network prediction. A general strategy is to first generate cores of known networks according to the knowledge in the KEGG reference pathways and then to extend the cores by searching for additional partners that are associated in the genome (e.g., genes in the same operon), the transcriptome (e.g., coexpressed genes), and the proteome (e.g., binding partners). The first step is called pathway reconstruction, which is basically the matching of genes in the genome and gene products in the pathway. To enable this matching, the genes in the genome must be assigned the ortholog identifiers according to, for example, sequence similarities and positional correlations of genes in other genomes. The second step is a more ambitious step, which can be formulated as a path computation problem in a graph or a set of binary relations. The path computation has been used to compute alternative enzymatic reaction pathways from a set of substrate-product relations (Goto et al. 1997). A similar strategy should be effective and it is being implemented in the BRITE database.

12.4.3 Gene Annotation by Pathway Reconstruction

When an organism-specific pathway is reconstructed by matching genes in the genome against KEGG reference pathways, a few genes are often missing in an otherwise complete pathway. Most of the cases can be solved by reexamining gene annotations and assignments of ortholog identifiers. The information about pathways and complexes imposes an additional constraint of completeness, which is extremely useful for interpreting sequence similarity scores, especially when many paralogs are present, because in general there is no predefined level of sequence similarity that can safely be extended to functional identity.

A case in point is the lysine biosynthesis pathway (Bono et al. 1998; Kanehisa 2000b), in which an aminotransferase gene was missing, as shown in figure 12.5. Here each box is an enzyme (gene product) with the EC number inside and the shading indicates that the corresponding gene is present in the genome. This pathway was biochemically determined in *E. coli*, and the gene names were assigned by genetic studies, as indicated alongside the boxes. However, when the complete genome sequences were determined, the *dapC* gene for succinyldiaminopimelate aminotransferase (EC 2.6.1.17) could not be found in *E. coli* or any other genomes. Recently, it was reported that *N*-acetylornithine aminotransferase (EC 2.6.1.11) in *E. coli*, which is encoded in *argD* and which functions in the arginine biosynthesis pathway, had a dual role of catalyzing the reaction by DapC as well (Ledwidge and Blanchard 1999).



Organism	Candidate genes *Escherichia coli argD*	*Bordetella pertussis dapC*
Escherichia coli	b3359 / b1748	b2379
Haemophilus influenzae	-	-
Helicobacter pylori	-	HP0624
Campylobacter jejuni	Cj0227	Cj0150c
Neisseria meningitidis	NMB1371	NMB0894
Bacillus subtilis	argD	yugH / ykrV
Mycobacterium tuberculosis	Rv3329	Rv0858c
Synechocystis	slr1022	sll0006
Aquifex aeolicus	aq_023	aq_273

Figure 12.5
The aminotransferase gene dapC in the lysine biosynthesis pathway had not been found in any of the completely sequenced genomes, but two recent reports found probable genes for this missing enzyme. The enclosed table summarizes homologs for these genes in different genomes.

Furthermore, *dapC* was found as part of the operon encoding *dapCDE* in *Bordetella pertussis* (Fuchs et al. 2000).

Aminotransferases form a family of paralogous proteins. It is impossible to predict substrate specificity from sequence similarity alone because the number of paralogs is different in different genomes and some aminotransferases must have dual roles. For example, aspartate aminotransferase (EC 2.6.1.1) and tyrosine aminotransferase (EC 2.6.1.5) are encoded by different genes with high sequence similarity in *E. coli*, but there is apparently no tyrosine aminotransferase gene in *Haemophilus influenzae*, and aspartate aminotransferase appears to function in the tyrosine pathway as well. In most genomes there are unassigned aminotransferases, especially those similar to aspartate aminotransferases, and *B. pertussis dapC* belongs to the aspartate aminotransferase subfamily. We have searched homologs of *E. coli argD* and *B. pertussis dapC* in the genomes of other organisms. The result shown in figure 12.5 has identified a homolog of *B. pertussis dapC* in *E. coli*, b1748, which is annotated as putative aminotransferase in the original database. It would be interesting to see if this gene product does have the DapC activity. Although the new findings did fill in the gaps in many genomes, some genomes are still unaccounted for, such as *H. influenzae*. It is

still possible that aspartate aminotransferase or its paralog takes care of the lysine pathway as well in some genomes.

12.5 Concluding Remarks

For all the genomes that have been sequenced, there is a considerable number of genes whose functions are not yet understood. The fraction of unknown genes varies in the genomes and also depends on the definition of function. As we have seen, the assignment of a general molecular function like an aminotransferase, a kinase, or an ABC transporter, does not tell much about a specific role of the gene in a cellular function. When this is considered, perhaps one-half to two-thirds of the genes are still unknown in most genomes. There have been attempts to systematically uncover functions of unknown genes in functional genomics experiments. Although such experiments may be useful to obtain a rough draft of gene functions, they are unlikely to provide detailed pictures of molecular interactions and pathways that are responsible for specific cellular functions. It is necessary to integrate with more accurate, traditional methods in biochemistry, molecular and cellular biology, and genetics. The KEGG resource should be useful for this integration.

We have limited our discussions to the level of molecular interactions, but there are still more issues concerning the association of genes and higher level biological phenomena, such as brain functions, diseases, and human behaviors, where cellular interactions and organism interactions must play more dominant roles (table 12.1). Although KEGG does not attempt to move up to such higher level phenomena, it contains information about, for example, disease classification and cell lineages in order to better understand underlying molecular phenomena.

The wiring information represented in KEGG pathway diagrams may appear to be static. However, there are two mechanisms to incorporate time- and space-dependent behaviors of the network of interacting molecules. One is the coloring mechanism used to generate organism-specific pathways. For example, the coloring of microarray hybridization data can be mapped onto the KEGG reference pathways and the time-course of gene expression changes can be followed by the changes in the coloring. The other mechanism is to simply draw additional pathway diagrams, each of which is considered to represent a snapshot of the dynamic change. KEGG is not suitable for simulating continuous behaviors of the cell because it does not contain any kinetic parameters. However, we still hope that KEGG will become useful to simulate perturbations to the cell, such as gene mutations and environmental changes, and their dynamic consequences.

Acknowledgements

The KEGG project is supported by the grants from the Ministry of Education, Science, Sports and Culture of Japan, the Science and Technology Agency of Japan, and the Japan Society for the Promotion of Science. The computational resources are provided by the Supercomputer Laboratory of the Institute for Chemical Research, Kyoto University.

References

Ashburner, M., Ball, C. A., Blake, J. A., Botstein, D., Butler, H., Cherry, J. M., Davis, A. P., Dolinski, K., Dwight, S. S., Eppig, J. T., Harris, M. A., Hill, D. P., Issel-Tarver, L., Kasarskis, A., Lewis, S., Matese, J. C., Richardson, J. E., Ringwald, M., Rubin, G. M., and Sherlock, G. (2000). Gene ontology: Tool for the unification of biology. *Nature Genet.* 25: 25–29.

Bono, H., Ogata, H., Goto, S., and Kanehisa, M. (1998). Reconstruction of amino acid biosynthesis pathways from the complete genome sequence. *Genome Res.* 8: 203–210.

Fuchs, T. M., Schneider, B., Krumbach, K., Eggeling, L., and Gross, R. J. (2000). Characterization of a bordetella pertussis diaminopimelate (DAP) biosynthesis locus identifies dapC, a novel gene coding for an N-succinyl-L,L-DAP aminotransferase. Bacteriology 182: 3626–3631.

Fujibuchi, W., Goto, S., Migimatsu, H., Uchiyama, I., Ogiwara, A., Akiyama, Y., and Kanehisa, M. (1998). DBGET/LinkDB: An integrated database retrieval system. *Pacific Symp. Biocomput. '98*, Altman, R. B., Dunker, A. K., Hunter, L., and Klein, T. E., eds. 683–694. Singapore: World Scientific.

Fujibuchi, W., Ogata, H., Matsuda, H., and Kanehisa, M. (2000). Automatic detection of conserved gene clusters in multiple genomes by graph comparison and P-quasi grouping. *Nucl. Acids Res.* 28: 4029–4036.

Goto, S., Bono, H., Ogata, H., Fujibuchi, W., Nishioka, T., Sato, K., and Kanehisa, M. (1997). Organizing and computing metabolic pathway data in terms of binary relations. *Pacific Symp. Biocomput. '97*, 175–186.

Goto, S., Nishioka, T., and Kanehisa, M. (1998). LIGAND: Chemical database for enzyme reactions. *Bioinformatics* 14: 591–599.

Kanehisa, M. (1997a). A database for post-genome analysis. *Trends Genet.* 13: 375–376.

Kanehisa, M. (1997b). Linking databases and organisms: GenomeNet resources in Japan. *Trends Biochem. Sci.* 22: 442–444.

Kanehisa, M. (2000a). *Post-genome Informatics*. Oxford: Oxford University Press.

Kanehisa, M. (2000b). Pathway databases and higher order function. *Adv. Protein Chem.* 54: 381–408.

Kanehisa, M., and Goto, S. (2000). KEGG: Kyoto encyclopedia of genes and genomes. *Nucl. Acids Res.* 28: 27–30.

Ledwidge, R., and Blanchard, J. S. (1999). The dual biosynthetic capability of N-acetylornithine aminotransferase in arginine and lysine biosynthesis. *Biochemistry* 38: 3019–3024.

Murzin, A. G., Brenner, S. E., Hubbard, T., and Chothia, C. (1995). SCOP: A structural classification of proteins database for the investigation of sequences and structures. *J. Mol. Biol.* 247: 536–540.

Ogata, H., Fujibuchi, W., Goto, S., and Kanehisa, M. (2000). A heuristic graph comparison algorithm and its application to detect functionally related enzyme clusters. *Nucl. Acids Res.* 28: 4021–4028.

Riley, M. (1993). Functions of the gene products of Escherichia coli. *Microbiol. Rev.* 57: 862–952.

Tatusov, R. L., Koonin, E. V., and Lipman, D. J. (1997). A genomic perspective on protein families. *Science* 278: 631–637.

13 Datamining: Discovering Information from Bio-Data

Limsoon Wong

13.1 Introduction

This chapter is an introduction to what has come to be known as datamining and knowledge discovery in the biomedical context. The major reason that datamining has attracted increasing attention in the biomedical industry in recent years is due to the increased availability of a huge amount of biomedical data and the imminent need to turn such data into useful information and knowledge. The knowledge gained can lead to improved drug targets, improved diagnostics, and improved treatment plans.

Datamining is the task of discovering patterns from large amounts of potentially noisy data where the data can be kept in regular relational databases or other forms of information repositories such as the flat text files commonly used by biologists. It is a very interdisciplinary subject, relying on ideas and developments in database systems, statistics, machine learning, data visualization, neural networks, pattern recognition, signal processing, and so on. More background on datamining is presented in section 13.2, where we describe the key steps of the knowledge discovery process, the diverse functionalities of datamining, and some popular datamining techniques.

Datamining has many functionalities, such as association analysis, classification, prediction, clustering, and trend analysis. The material in this chapter is presented from the classification perspective, where emphasis is placed on basic techniques for uncovering interesting factors that differentiate one class of samples from a second class of samples. Specifically, the chapter describes datamining techniques for the classification of MHC-binding peptides and diabetes clinical study data. These two types of data are chosen because they are very different in nature and thus require very different datamining techniques.

The classification of MHC-binding peptides is described in section 13.3. It is a target discovery problem in computational immunology. It is an illustration of the application of an artificial neural network to the classification of noisy homogeneous biomedical data. Our description is based on a collaboration (Brusic and Zeleznikow 1999) between Kent Ridge Digital Labs and the University of Pittsburgh.

The classification of diabetes clinical study data is described in section 13.4. It is a problem of forecasting the onset of diabetes. It is an illustration of the application of an idea known as emerging patterns to the classification of heterogenous biomedical data. Our description is based on a collaboration (Dong et al. 1999) between Kent Ridge Digital Labs and the University of Melbourne.

13.2 Datamining Background

Datamining is a natural evolution of information technology along the path of data collection, database creation, database management, and data analysis and interpretation (Han and Kamber 2000). Here, we briefly explain various aspects of datamining and knowledge discovery in general terms.

13.2.1 Process

The knowledge discovery process can be broken down into six stages (Adriaans and Zantinge 1996): data selection, cleansing, enrichment, coding, datamining, and reporting.

The first stage of the knowledge discovery process is collection and selection. In the case of the MHC-binding peptide example, it is the collection of information on what peptides are known to bind or not bind which MHC molecules. In the case of the diabetes example, it is the collection of certain clinical information from a select group of diabetic and non-diabetic patients.

The second stage is a data cleansing process to remove noise or irrelevant data. An important element of this process is the de-duplication of data records to produce a non-redundant dataset. For example, the same MHC-binding peptide information may be reported in two separate papers. Another important element of this process is the normalization of data records to deal with pollution caused by the lack of domain consistency. This type of pollution is particularly damaging because it is hard to trace. For example, MHC-binding peptide information reported in a paper might be wrong due to a variety of experimental factors. In fact, Schoenbach et al. (2000b) made a detailed study of swine MHC sequences and found that of the 163 records they examined, there were 36 critical mistakes. Similarly, clinical records from different hospitals may use different terminologies, different measures, capture information in different forms, or use different default values to fill in blanks. As another example, gene expression experiments under similar conditions may produce different data because the overall intensity of a DNA chip can vary substantially from chip to chip.

The third stage, enrichment, is the acquisition of extra information that can be integrated into the existing data. For example, disease demographic data or linkage data in the case of clinical records, or known biological pathway information relating to genes on a DNA chip.

The fourth stage is coding, where data are transformed or consolidated into forms appropriate for datamining. In the case of clinical records, this might be the trans-

formation of the absolute age of a patient into groupings such as "young," "middle aged," and "old." In the case of MHC-binding peptides, this might be the transformation of the MHC-binding affinity into groups such as "nonbinder," "weak binder," "moderate binder," and "strong binder." Sometimes, this coding step is performed using an automatic discretization algorithm such as the entropy method (Kohavi and Sahami 1996).

The fifth stage, data mining, is the phase of real discovery. It is an essential process where intelligent methods are applied in order to extract data patterns. We discuss it in greater detail shortly. The last stage is reporting, where visualization and knowledge representation techniques are used to present the mined knowledge to the user, or where a prediction system is produced.

13.2.2 Functionalities

In general, datamining tasks can be split into two categories: descriptive and predictive (Han and Kamber 2000). Descriptive datamining tasks characterize the general properties of the data. Predictive datamining tasks perform inference on the current data in order to make predictions. We briefly touch on the main varieties of datamining tasks and functionalities below.

Classification is the process of finding a set of models that describe and distinguish between two or more data classes or concepts. The model is derived by analyzing a set of training data that have been explicitly labeled with the classes that they belong to. The model is then used to predict the class of objects whose class label is unknown. In the rest of this chapter, we describe in detail two examples of classification and prediction and also briefly survey a number of other examples of classification and prediction in molecular biology.

Cluster analysis, by contrast, is used in situations where the training data do not have any known class labels. The purpose of clustering is to generate class labels for the data. The data objects are typically clustered so that objects within a cluster have a high similarity to each other but are very dissimilar to objects in other clusters. Much work (Ben-Dor 1999; Eisen et al. 1998) on analyzing gene expression data belong to this category of datamining tasks.

Outlier analysis deals with objects that do not comply with the general behavior of a model of the data. Most datamining applications discard outliers. However, in some applications, the rare events can be more interesting than the more regularly occurring ones—for example, the detection of new particles in nuclear accelerator experiments.

Trend analysis describes and models regularities or trends for objects whose behavior changes over time. The distinctive features of such an analysis include time-series data analysis, periodicity pattern matching, and clustering of time-related data.

The inference of gene relationships from large-scale temporal gene expression patterns (D'haeseleer et al. 1998) is an example of this topic.

Association analysis is the discovery of rules showing attribute-value conditions that occur frequently together in a dataset, such as the co-expression of genes. It is not difficult to develop algorithms for detecting association rules in a large database. The problem is that such an algorithm often returns so many associations that it is difficult to distinguish interesting associations from uninteresting ones. The idea of emerging patterns, to be seen later in the classification of diabetes clinical data, is one method for separating interesting information from uninteresting information.

13.2.3 Techniques

Many techniques have been used in datamining. We now briefly survey some data-mining techniques that have been successfully used for classification in the biomedical context.

The most popular classification technique is the idea of decision tree induction, where a dataset is recursively partitioned into discrete subgroups based on the value of an attribute in the dataset. The remaining attributes in the dataset are selected as to whether or not they provide a predictive segregation of the remaining data for different values of the classification variable. The final result is a set of series of splits on values of the attributes, each series of which leads to a classification value. Algorithms for decision tree induction include CART (Breiman et al. 1984), ID3 (Quinlan 1986), C4.5 (Quinlan 1992), SLIQ (Mehta et al. 1996), FACT (Loh and Vanichsetakul 1988), QUEST (Loh and Shih 1997), PUBLIC (Rastogi and Shim 1998), CHAID (Kaas 1980), ID5 (Utgoff 1988), SPRINT (Shafer et al. 1996), and BOAT (Gehrke et al. 1999). This group of algorithms are most successful for analysis of clinical data and for diagnosis from clinical data. Some examples are diagnosis of central nervous system involvement in hematooncologic patients (Lossos et al. 2000), prediction of post-traumatic acute lung injury (Rainer et al. 1999), identification of acute cardiac ischemia (Selker et al. 1995), prediction of neurobehavioral outcome in head-injury survivors (Temkin et al. 1995), diagnosis of myoinvasion (Longacre et al. 1995), and so on.

Another popular classification technique is Bayesian classification. It is based on the Bayes theorem, $P(H \mid X) = P(X \mid H) \times P(H)/P(X)$, which provides a way to derive the posterior probability $P(H \mid X)$ that a hypothesis H holds given a sample X from the prior probabilities $P(H)$, $P(X)$, and $P(X \mid H)$. Note that many Bayesian classifiers make the simplifying assumption that the effect of an attribute value of a given class is independent of the values of the other attributes. Nevertheless, these Bayesian classifiers are comparable in performance to decision tree and neural net-

work classifiers in many applications. (More information on algorithms for Bayesian classifiers can be obtained in Baldi and Brunak 1999; Duda and Hart 1973; Mitchell 1997; John 1997; Heckerman 1996; Jensen 1996; Russell et al. 1995; and Lauritzen 1995). Some example applications of Bayesian classifiers in the biomedical context are mapping of a locus controlling a genetic trait (Ghosh and Majumder 2000), screening for macromolecular crystallization (Hennessy et al. 2000), classification of cNMP-binding proteins (McCue et al. 2000), prediction of carboplatin exposure (Huitema et al. 2000), prediction of prostate cancer recurrence (Demsar et al. 1999), prognosis of femoral neck fracture recovery (Kukar et al. 1996), prediction of protein secondary structure (Kasif and Delcher 1998; Stultz et al. 1997; Arnold et al. 1992), and so on.

Related to the Bayesian classifiers are the hidden Markov models, or HMMs. An HMM is a stochastic generative model for sequences defined by a finite set S of states, a finite alphabet A of symbols, a transition probability matrix T, and an emission probability matrix E. The system moves from state to state according to T while emitting symbols according to E. In an n-th order HMM, the matrices T and E depend on all n previous states. (More detailed introduction to HMMs can be found in Baldi and Brunak 1999; Krogh 1998; Durbin et al. 1998; and Eddy 1996). HMMs have been applied to a variety of problems in sequence analysis, including protein family classification and prediction (Bateman 1999; Baldi and Chauvin 1994; Krogh et al. 1994), tRNA detection in genomic sequences (Lowe and Eddy 1997), Methylation guide snoRNA screening (Lowe and Eddy 1999), gene finding and gene structure prediction in DNA sequences (Borodovsky et al. 1995; Borodovsky and McIninch 1993; Baldi et al. 1997; Krogh 1998; Salzberg et al. 1998), protein secondary structure modeling (Di Francesco et al. 1997), promoter recognition (Yada et al. 1996; Pedersen et al. 1996), and so on.

Artificial neural networks are another important approach to classification that have a high tolerance to noisy data. A more detailed introduction to them is presented in a later section on the prediction of peptide binding to MHC molecules. (Other useful presentation of artificial neural network algorithms can be found in Rumelhart et al. 1986; Baldi and Brunak 1999; and Chauvin and Rumelhart 1995). Successful applications of artificial neural networks in the biomedical context include protein secondary structure prediction (Riis and Krogh 1996; Rost and Sander 1994; Qian and Sejnowski 1988), signal peptide prediction (Claros et al. 1997; Nielsen et al. 1997; Emanuelsson et al. 1999), gene finding and gene structure prediction (Uberbacher and Mural 1991; Snyder and Stormo 1995), protein translation initiation site recognition (Pedersen and Nielsen 1997), T-cell epitope prediction (Honeyman et al. 1998), RNA secondary structure prediction (Steeg 1993), toxicity prediction (Burden and

Winkler 2000), disease diagnosis and outcome prediction (Vriesema et al. 2000; Scott et al. 2000; Turton et al. 2000), and so on.

Support vector machines or SVMs are a new approach to the classification problem that has clear connections to statistical learning theory. They differ radically from approaches such as artificial neural networks. In particular, SVM training always finds a global minimum. A SVM is largely characterized by the choice of its kernel function. Thus SVMs connect the problem they are desgined for to a large body of existing research on kernel-based methods. (For a detailed discussion of SVMs, see Raudys 2000; Vapnik 1995; and Burges 1998). Some recent applications of SVM in the biomedical context include protein translation initiation site recognition (Zien et al. 1999), protein homology detection (Jaakkola et al. 2000), microarray gene expression data classification (Brown et al. 2000), breast cancer diagnosis (Mangasarian et al. 1995; Friess et al. 1998), and so on.

Let us now embark on our datamining examples on the classification of MHC-binding peptides and diabetes clinical data.

13.3 Short Peptides

Short peptides are a special case of protein sequence data. They are often the key to the functional role of protein sequences such as active sites, binding core, and so on. Their short length makes it possible to perform certain types of analysis on them that are not computationally feasible on long protein sequences. In this section, we describe a general method called "artificial neural networks" that has worked well on a broad class of classification and prediction problems involving short peptides.

13.3.1 Problem

The immune system employs two means of recognition: soluble antibodies and T-cell receptors. T-cell mediated immunity is the more subtle of the two and is further divided into those associated with class-I versus class-II MHC molecules. Killer T-cells continually scan the surface of all cells and destroy those with foreign markings that came from short peptides—derived from cytosolic proteins—bound to class-I MHC molecules. Helper T-cells scan cell surfaces for short peptides—derived from proteins internalized by endocytosis—bound to class-II MHC molecules; association of a foreign peptide to a class-II MHC molecule signals that a cell has encountered a pathogen and serves as a call for help (Stryer 1995). Peptides that induce immune response are called immunogenic peptides or T-cell epitopes. T-cell epitopes are targets for the discovery of vaccine and immunotherapeutic drug components.

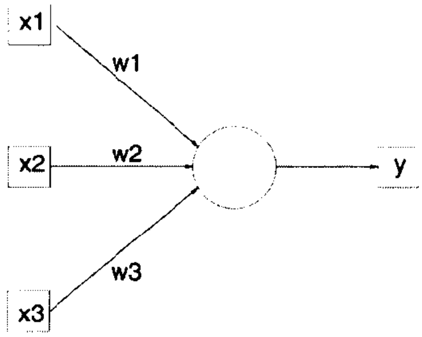


Figure 13.1
A artificial neural network computing element.

We use the work of Brusic and Zeleznikow (1999) and Brusic et al. (1998b) as an example classification problem in analysing short peptides. Brusic employs an artificial neural network (Patterson 1996), implemented using the PlaNet package (Miyata 1991), for the identification of T-cell epitopes from melanoma antigens. That is, given a short peptide derived from a melanoma antigen, his artificial neural network classifies it according to its binding affinity to particular MHC molecules. Brusic's work scans four melanoma-related antigens (MART-1, gp100/pmel17, MAGE-3, and MAGE-6) to identify short peptides that bind HLA-DR4, a class II MHC molecule. Brusic's artificial neural network is trained from a set of 602 HLA-DR4 binding peptides and 713 nonbinders drawn from MHCPEP (Brusic et al. 1998a), experimental binding data (Hammer et al. 1994), and FIMM (Schoenbach et al. 2000a). Each training sample has an associated binding affinity that we refer to as the "targeted output" for that training sample. In the rest of this section, we supply the technical details.

13.3.2 Solution

Artificial neural networks are networks of highly interconnected neural computing elements that have the ability to respond to input stimuli and to learn to adapt to the environment. Although the architecture of neural networks differ in several characteristic ways, a typical artificial neural network computing element is a comparator that produces an output when the cumulative effect of the input stimuli exceeds a threshold value.

Figure 13.1 depicts a single computing element. Each input x_i has an associated weight w_i, which acts to either increase or decrease the input signal to the computing element. The computing element behaves as a monotonic function f producing an output $y = f(net)$, where net is the cummulative input stimuli to the neuron. The number net is usually defined as the weighted sum of the inputs:

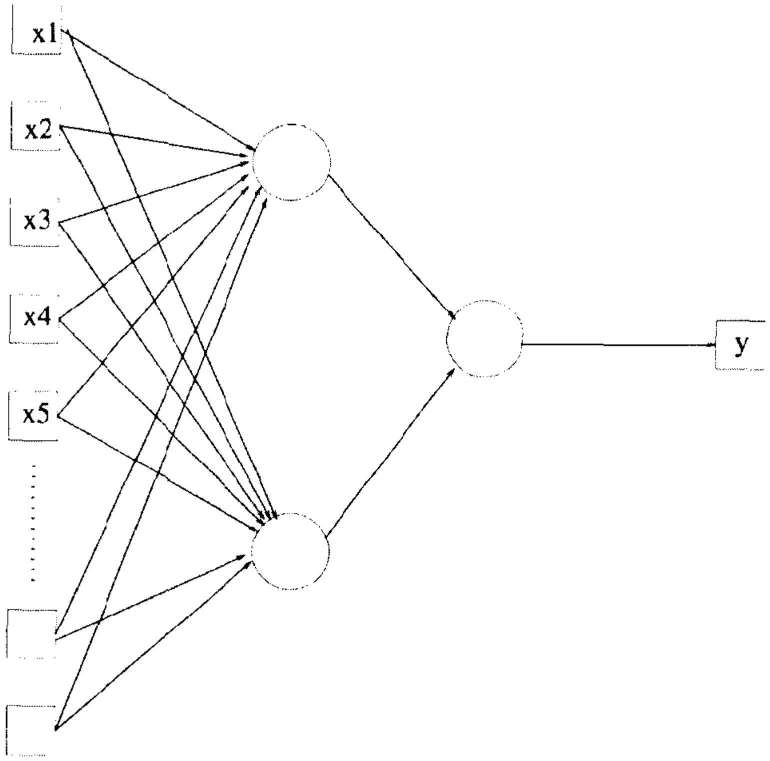


Figure 13.2
A fully connected feedforward artificial neural network with two layers of computing elements.

$$net = \sum_i x_i w_i$$

and the function f is usually defined as a sigmoid:

$$f(net) = \frac{1}{1 + e^{-net}}$$

Such computing elements can be connected in multiple layers into an artificial neural network. Figure 13.2 depicts a fully connected feed-forward artificial neural network with two layers of computing elements. The output from the two hidden computing elements in the first layer are fed as inputs into the computing element at the second layer. The network is used for classifiction decision as follows. The inputs x_i are fed into the network. Each computing element at the first layer produces its corresponding output, which is fed as input to the computing elements at the next layer. This process continues until an output is produced at the computing element at the final layer.

The artificial neural network used by Brusic and Zeleznikow (1999) has just two layers, with two computing elements at the first layer and one computing element at the second layer. It is also fully connected in the sense that each input x_i is connected to both computing elements at the first layer, both of which are in turn connected to the computing element at the second layer. From previous experience (Brusic et al. 1998b), other architectures for the artificial neural network are possible but are not expected to yield significant differences. A short peptide derived from an antigen is converted into input values x_i according to a sparse coding scheme, where each amino acid in the short peptide is represented as a string of 20 bits. For example, Alanine is represented by "10000000000000000000" and Cysteine is represented by "01000000000000000000." From previous experience (Brusic et al. 1994), other representations are possible but are not expected to produce significantly different results. Because HLA-DR4 is a class-II MHC molecule, a short peptide will bind it through a 9-mer core with the rest of the peptide flanking outside the MHC cleft (Rammensee et al. 1995). In other words, we need to consider only 9-mers and thus there are $9 \times 20 = 180$ bits to be used as inputs per peptide. So Brusic's artificial neural network has 180 inputs, $x_1, \ldots, x_{180}$. An input sample is classified as HLA-DR4 binding if the final output exceeds a threshold, and as non-HLA-DR4 binding otherwise.

The details are in how the weights on the links between the inputs and computing elements are chosen. These weights are learned from training data using the error back-propagation method (Rumelhart et al. 1986). To describe the error back-propagation method, we need to introduce some notations. Let v_{ij} denote the weight on the link between x_i and the jth computing element of the first layer in the artificial neural network. Let w_j denote the weight on the link between the jth computing element of the first layer and the computing element of the last layer. Let z_j denote the output produced by the jth computing element of the first layer. Then the output y produced by the artificial neural network for a given training sample is given by

$$y = f\left(\sum_j w_j f\left(\sum_i x_i v_{ij}\right)\right)$$

This y may differ from the targeted output t for that particular training sample by an error amount Δ. We need a method to reduce this error through an adjustment of the weights v_{ij} and w_j. This is accomplished by adjusting the weights in proportion to the negative of the error gradient. For mathematical convenience, the squared error E can be defined as

$$E = \frac{(t - y)^2}{2}$$

In finding an expression for the weight adjustment, we must differentiate E with respect to weights v_{ij} and w_j to obtain the error gradients for these weights. Applying the chain rule a couple of times and recalling the definitions of y, z_j, E, and f, we derive

$$
\begin{aligned}
\frac{\delta E}{\delta w_j} &= \frac{\delta E}{\delta \sum_j w_j f(\sum_i x_i v_{ij})} \frac{\delta \sum_j w_j f(\sum_i x_i v_{ij})}{\delta w_j} \\
&= \frac{\delta E}{\delta \sum_j w_j f(\sum_i x_i v_{ij})} f\left(\sum_i x_i v_{ij}\right) \\
&= \frac{\delta E}{\delta y} \frac{\delta y}{\delta \sum_j w_j f(\sum_i x_i v_{ij})} f\left(\sum_i x_i v_{ij}\right) \\
&= (t - y) f'\left(\sum_j w_j f\left(\sum_i x_i v_{ij}\right)\right) f\left(\sum_i x_i v_{ij}\right) \\
&= (t - y) f'\left(\sum_j w_j z_j\right)(z_j) \\
&= (\Delta)(y)(1 - y)(z_j)
\end{aligned}
$$

The last step follows because f is a sigmoid and thus $f'(x) = f(x)(1 - f(x))$. Then the adjustment Δ_{w_j} to w_j is defined as below, where η is a fixed learning rate.

$$\Delta_{w_j} = -\eta \frac{\delta E}{\delta w_j} = -\eta(\Delta)(y)(1 - y)(z_j)$$

However, for the weights v_{ij}, we do not have a targeted output to compute errors, so we have to use the errors Δ_{w_j} as surrogates and apply a similar derivation to obtain

$$\Delta_{v_{ij}} = -\eta(\Delta_{w_j})(z_j)(1 - z_j)(x_i)$$

The above derivation provides the adjustments on the weights for one training sample. An "epoch" in the training process is a complete iteration through all training samples. At the end of an epoch, we compute the total error of the epoch as the sum of the squares of the Δ of each sample. If this total error is sufficiently small, the training process terminates. If the number of epochs exceeds some predefined limit, the training process also terminates.

An artificial neural network model for HLA-DR4 binding was built as described above using 602 HLA-DR4 binding peptides and 713 nonbinders. The model was then used to process a pool of 1,374 peptides from MART-1, gp100/pmel17, MAGE-

3, and MAGE-6. Thirty candidate binders identified by the aritificial neural network were synthesized and tested for T-cell responses and DR4 binding. Eighteen novel T-cell epitopes were confirmed, giving a $18/30 = 60\%$ success rate in predicting T-cell epitopes.

13.3.3 Remarks

A systematic experimental study of a protein antigen for T-cell epitopes would include the following steps. First, a large number of overlapping peptides spaning the length of the protein must be synthesized. Second, each of these peptides must be tested in binding assays. Typically, much less than 5 percent of peptides emerge from these assays as binders. Third, each of those peptides that bind must be further tested in T-cell assays to confirm immunogenicity. Typically much less than 1 percent of the original peptides emerge through these assays as immunogenic. Such an exhaustive study would be prohibitively expensive and time consuming.

A highly accurate method of predicting the binding of peptides to MHC molecules enables binding assays to be skipped or significantly reduced (Gulukota 1998; Brusic and Zeleznikow 1999). Given the protein sequence of the antigen, all overlapping peptides spanning the sequence are derived by computer. Each peptide is then classified using the MHC-peptide binding prediction method described earlier. Only those peptides classified by this method as a MHC-binder proceed to be tested in T-cell assays for their immunogenicity. Only for those peptides tested positive in T-cell assays do we perform binding assays. This new procedure would reduce the number of binding assays dramatically.

There are several alternatives to the artificial neural network approach described here. The simplest is the peptide-binding motif approach; many motifs have been proposed for various MHC molecules (Rammensee et al. 1995). These motifs usually indicate primary anchor positions and amino acids acceptable as primary anchors. The more sophisticated motifs usually take the form of a quantitative matrix that specifies the observed frequencies of different amino acids in each position. Their sensitivity and specificity levels are weaker than the artificial neural network approach (Brusic and Zeleznikow 1999) when there is a large amount of training data available. However, when only a small amount of training data is available, a quantitative matrix may be more useful. Molecular modeling has also been used for the prediction of peptide binding to MHC molecules (Rognan et al. 1994). Nevertheless, the accuracy of molecular models needs to be improved before they can be used for new predictions. Additional methods for predicting MHC-peptide binding are described in several newly published articles (Borras-Cuesta et al. 2000; Raddrizzani and Hammer 2000; Savoie et al. 1999). Based on their reported sensitivity and precision figures, an

artificial neural network-based methodology, as reported by Brusic et al. (1998b) and described above, appears to be more accurate.

13.4 Clinical Records

The clinical record of a patient is a more general type of data. The characteristics of clinical records are very different from those of protein sequences or gene expression results. The most important difference is perhaps the heterogeneity in the attributes of a clinical record. The meaning of one attribute in a clinical record can be very different from another attribute. For example, a clinical record may have an attribute recording the age of the patient and an attribute recording the blood pressure of the patient. In contrast, in a gene expression record, the attributes correspond to genes and the value of each attribute is the expression level of the corresponding gene, and thus all attributes have the same kind of values. Furthermore, clinical records kept for different studies or by different organizations can differ significantly in the information they capture. Therefore, the general analysis of patient clinical records calls for datamining methods that make fewer assumptions and interpretations of the data. In this section, we describe a general method called "classification by aggregating emerging patterns," or CAEP for short, that has worked well on clinical records and other similar types of data (Dong et al. 1999).

13.4.1 Problem

Classification is an interesting problem in analyzing patient clinical records. Suppose we have a population of patients (more generally, samples or instances) that are divided into two groups. For example, we can divide our population of patients with a particular type of cancer into those who responded to a treatment versus those who did not. As another example, we can divide a population of samples into those who showed signs of a disease versus those who did not. The task of a classifier is to discover the factors that differentiate the two groups and to find a way to use these factors to predict to which group a new patient should belong.

As our example, we use the Pima Indians dataset (Smith et al. 1988). The data were collected by the U.S. National Institute of Diabetes and Digestive and Kidney Diseases from a population of 768 women who were at least 21 years old, of Pima Indian heritage, and living near Phoenix, Arizona. They were tested for diabetes according to World Health Organization criteria; that is, the two-hour post-load plasma glucose was at least 200 mg/dl at any survey examination or during routine medical care. For each patient, eight attributes were obtained: (1) number of times

pregnant, (2) plasma glucose concentration, (3) diastolic blood pressure, (4) triceps skin fold thickness, (5) two-hour serum insulin, (6) body mass index, (7) diabetes pedigree function, and (8) age. Obviously these attributes are heterogeneous; for example, age is in years and diastolic blood pressure in mm Hg. Also, there is a size bias in this dataset in that it contains 29 percent diabetic instances and 71 percent non-diabetic instances.

The CAEP method, in this particular example, builds a classifier using the Pima Indian dataset. Given a new patient record with these eight attributes, the classifier predicts if the patient has diabetes according to World Health Organization criteria.

13.4.2 Solution

The CAEP relies on the recently proposed idea of *emerging patterns* (Dong and Li 1999). An emerging pattern is a pattern whose frequency increases significantly from one class of data to another class of data. For example, the pattern {odor = none, stalk-surface-below-ring = smooth, ring-number = 1} in the description of mushrooms by the Audubon Society (Lincoff 1981) is a typical emerging pattern. Its frequency increases from 0.2 percent in the poisonous case to 57.6 percent in the edible case, at a growth rate of 57.6/0.2 = 288. Each emerging pattern can have very strong power for differentiating the class membership of some instances. For example, with odds of 99.6 percent—we will show how this is derived later—a mushroom that contains the above pattern is edible. The differentiating power of an emerging pattern is a function of its growth rate and frequency. However, an individual emerging pattern may only be able to classify a small number of instances. Thus it may have poor overall accuracy if it is used by itself on all instances. So to build an accurate classifier, it is necessary to make use of multiple emerging patterns. The CAEP discovers, for each class of data, all the emerging patterns of that class satisfying some threshold on frequency and growth rate. The differentiating power of these emerging patterns are then summed and normalized. The CAEP then chooses the class with the largest normalized score as the winner. We now proceed to describe the CAEP in detail.

We need to discretize of the dataset into a binary one. The value range of each attribute is discretized into a number of intervals using the entropy method (Kohavi and Sahami 1996). Each (*attribute*, *interval*) pair is called an *item* in the binary database. An instance t in the raw dataset is then thought of as a set of items such that an item (A, v) is in t if and only if the value of the attribute A of t is within the interval v. We use the term *itemset* to refer to t under this representation. In an item (A, v) in an itemset, the value v is not allowed to be a null value. As a consequence, missing data that often plague clinical records are conveniently taken care of.

The (binary) Pima Indian dataset is divided into two sets that we denote D_1 and D_2, corresponding to diabetic and non-diabetic class of instances. We also use the notation D to denote either one of D_1 or D_2 and D' to denote the other set. Let I be the set of all possible items and $X \subseteq I$ be an itemset. The *support* of X in D is defined as

$$supp_D(X) = \frac{|\{t \in D \mid X \subseteq t\}|}{|D|}$$

The *growth rate* of X in D is defined as

$$grow_D(X) = \frac{supp_D(X)}{supp_{D'}(X)}$$

An *emerging pattern* of class D is an itemset that has "large" growth rate in D. Here "large" is an application-dependent threshold chosen by CAEP automatically based on the training dataset.

Each emerging pattern can differentiate the class membership of a fraction of instances that contain that emerging pattern. This differentiating power is derived from the difference between its support in the two sets. Suppose an instance t contains a particular emerging pattern X of class D. What is the likelihood that t belongs to class D? If D and D' are roughly equal in size, this likelihood is

$$likelihood_D(X) = \frac{supp_D(X)}{supp_D(X) + supp_{D'}(X)}$$

Now, using $grow_D(X) = supp_D(X)/supp_{D'}(X)$, we have

$$likelihood_D(X) = \frac{grow_D(X) * supp_{D'}(X)}{grow_D(X) * supp_{D'}(X) + supp_{D'}(X)} = \frac{grow_D(X)}{grow_D(X) + 1}$$

If D and D' differ significantly in size, the supports should be replaced by counts of X in D and D'; and thus

$$likelihood_D(X) = \frac{supp_D(X) * |D|}{supp_D(X) * |D| + supp_{D'}(X) * |D'|} = \frac{grow_D(X) * |D|}{grow_D(X) * |D| + |D'|}$$

Now consider a specific emerging pattern X of class D, say the diabetic class, having a moderate growth rate in D, say 3, and likelihood, say 75 percent, that appears in a relatively large number of instances in D, say 30 percent. Then X appears in $30\%/3 = 10\%$ of instances in D'. If we were to use X as our sole clue for class prediction, the achieved sensitivity for class D would be 30 percent and specificity would

be 55 percent. (Recall that the number of non-diabetic samples is $545 = 71\% * 768$ and diabetic samples is $223 = 29\% * 768$. Furthermore, sensitivity is defined as the ratio of the number of correctly predicted diabetic instances to the number of diabetic instances and specificity is defined as the ratio of the number of correctly predicted diabetic instances to the number of predicted diabetic instances.) Hence the overall accuracy, defined as the percentage of instances correctly classified, would be 72 percent.

In spite of the relatively high accuracy in using this particular emerging pattern as the sole clue for class prediction, there is a crucial problem. It could only identify 30 percent of the diabetic patients. This level of sensitivity cannot be considered acceptable as it is much more important to recognize a diabetic person than a non-diabetic one. Furthermore, in real-life clinical data, there may be no emerging pattern that has more than 3–5 percent support. This situation calls for a more innovative use of emerging patterns, where we combine the strength of several emerging patterns to produce a good classifier.

Given a test instance t, we let all emerging patterns of the class D that t contains contribute to the decision of whether t belongs to D. The advantage is that this way more cases can be covered because different emerging patterns can complement each other in their applicable populations. In order to combine emerging patterns, we need a scoring method. Given an instance t and a set E of emerging patterns of a class D, the score of t for D is defined as

$$score(t, D) = \sum_{X \subseteq t,\, X \in E} likelihood_D(X) * supp_D(X)$$

It is tempting to classify a test instance t as D if $score(t, D) > score(t, D')$, and as D' otherwise. However, such a simple-minded approach is not robust when the numbers of emerging patterns in D and D' differ significantly. This situation is quite common in clinical records where one of the classes have more random distribution of values and consequently fewer emerging patterns. In order to be more robust, the score should be normalized by dividing with a score at a fixed percentile of the instances in each class. More specifically, for each class D, a base score $base_score(D)$ should be found for D and then the normalized score is defined as

$$norm_score(t, D) = \frac{score(t, D)}{base_score(D)}$$

Now given a new instance t, we classify it as a member of class D if $norm_score(t, D) > norm_score(t, D')$ and as D' otherwise.

It remains to choose $base_score(D)$. It can be chosen as the median of scores $score(t, D)$ over all $t \in D$. In this case, exactly half of the instances in D have score greater than $base_score(D)$ and half exactly less than it. In practice, other percentiles between 50–85 percent produce roughly similar results. However, one should avoid percentiles at the extreme ends, say 3 percent, because clinical records are likely to contain some outliers and choosing such percentiles would give these outliers too much influence.

We compare the performance of the CAEP method described above with two state-of-the-art methods, C4.5 (Quinlan 1992), and CBA (Liu et al. 1998) under 10-fold cross validation. The accuracy of C4.5 is 71.1 percent. The accuracy of CBA is 73.0 percent. The accuracy of CAEP is higher at 75.0 percent. In fact, its sensitivity and specificity for the diabetic class are 70.5 percent and 63.3 percent, and for the non-diabetic class are 77.4 percent and 83.1 percent.

It is also reasonable to expect that the emerging patterns used for such highly accurate classification can give insight into important factors of diabetes. Unfortunately, the CAEP method described above produces a very large number of emerging patterns in general. It produces about twenty-three hundred emerging patterns with the Pima Indian dataset. It is too time consuming for a medical expert to analyze such a large number of emerging patterns. This brings us to add a reduction step.

The CAEP method reduces the number of emerging patterns based on the following factors: the strength of emerging patterns, the relationships between emerging patterns, and the difference between their supports and growth rates. The main idea is to prefer strong emerging patterns over their weaker relatives. Let ρ be a new growth rate threshold chosen that is larger than the initial threshold. Let $X_1 >_D X_2$ means X_1, X_2 are both emerging patterns of class D and X_1 is preferrable to X_2; and it is defined as the relation such that

$$X_1 >_D X_2 \quad \text{iff (a) } X_1 \subseteq X_2 \text{ and } grow_D(X_1) > grow_D(X_2)$$
$$\text{or (b) } X_1 \subseteq X_2 \text{ and } supp_D(X_1) \gg supp_D(X_2) \text{ and } grow_D(X_1) > \rho$$

The motivation for this definition is as follows. If $X_1 >_D X_2$ holds because of (a), then X_1 covers more cases than X_2—as $X_1 \subseteq X_2$—and also has stronger differentiating power—as $grow_D(X_1) > grow_D(X_2)$. The case where $X_1 >_D X_2$ holds because of (b) is more subtle. A typical situation captured by it is when X_2 is an emerging pattern with infinite growth rate but with very small support, whereas X_1 is an emerging pattern with lesser growth rate but has a much larger support, say 30 times more. In such a situation, X_1 is preferred because it covers many more cases than X_2, provided X_1 has a sufficiently high differentiating power—as guaranteed by the condition $grow_D(X_1) > \rho$. To illustrate this point, suppose $supp_D(X_1) = 100\%$, $grow_D(X_1) =$

22.25%, $supp_D(X_2) = 3\%$, and $grow_D(X_2) = \infty$. Clearly, X_1 is more useful than X_2, as it covers 33 times more instances and its associated likelihood, $22.25/(22.25 + 1) = 95.7\%$, is also very close to that of X_2.

The reduced set of emerging patterns of D is simply any maximal antichain of this relation. That is, for any X_1 and X_2 in the reduced set of emerging patterns, it is the case that neither $X_1 >_D X_2$ nor $X_2 >_D X_1$; and for any emerging pattern X_1 not in this set, it is the case that there is some emerging pattern X_2 in the reduced set such that $X_2 >_D X_1$.

Then *norm_score*, *score*, and *base_score* are all redefined in terms of the reduced set of emerging patterns. Now, given a new instance, we classify it as before: assign it to the class with the larger *norm_score*. We investigate the performance of the CAEP method with reduction of emerging patterns under 10-fold cross validation. As expected, the number of emerging patterns it produces is about sixteen hundred, which is considerably less than the twenty-three hundred emerging patterns without reduction. Its accuracy (75.1 percent), sensitivity for the diabetic class (69.0 percent), specificity for the diabetic class (64.1 percent), sensitivity for the non-diabetic class (78.4 percent), and specificity for the non-diabetic class (82.6 percent) are comparable to those without reduction of emerging patterns. Thus there is no loss in prediction performance using reduction of emerging patterns.

Let us summarize the steps of the CAEP method. It has two major types of inputs. The first input is the "training" dataset, which is the Pima Indian dataset set in our example. The second input is the "test" dataset, which contains test instances that we want to classify as diabetic or non-diabetic in this case. The CAEP method goes through two phases. The first phase is called the training phase and consists of discretizing the training dataset; datamining the discretized dataset for emerging patterns; computing the support, growth rate, and likelihood of each of these emerging patterns with respect to each of the classes; and computing the base score for each of the classes. The second phase is called the prediction phase and consists of computing the score and normalized score of each of the test instances with respect to each class, and making a prediction for each test instance. The training phase is the time-consuming part of the CAEP method and may take an hour or more. However, it needs to be done just once. Its results can be reused for many predictions, each of which typically takes much less than a second.

13.4.3 Remarks

In addition to the Pima Indian dataset, we have also tested the CAEP method on a large number of benchmark datasets. The CAEP method has very good predictive

accuracy on all data sets we have tested (Dong et al. 1999). It gives better accuracy than previous classification algorithms such as C4.5 (Quinlan 1992) and CBA (Liu et al. 1998) in general. The CAEP is highly accurate and is usually equally accurate on all classes, even if their proportions are unbalanced. Being equally accurate on all classes is very useful for many applications, where there is a dominant class and a minority class and the main purpose of classification is to accurately predict instances of the minority class. For example, in a preliminary diagnostic situation, one should err on the cautious side and call for further diagnostic tests for the gray cases.

The CAEP is rather different from previous classifiers because of the novelty of the emerging pattern idea. To arrive at a classification decision, the CAEP uses a set of emerging patterns and each emerging pattern corresponds to a multi-attribute test. Most previous classifiers such as C4.5 (Quinlan 1992) consider only one test on one attribute at a time. A few exceptions such as CBA (Liu et al. 1998) consider only one multi-attribute test to make a decision. It is also different from boosting (Schapire 1990), which manipulates training data to generate different classifiers and then aggregate the votes of these classifiers. In the CAEP case, although the emerging patterns are easy to determine, they are too weak to serve as individual classifiers.

There is, however, some similarity between the CAEP and Bayesian prediction. Let an instance t be viewed as a pattern. Bayesian prediction would classify t as belonging to a class C_k, where the probability $P(C_k) \times P(t \mid C_k)$ is largest among the classes. The optimal Bayesian classifier needs to "know" the probability $P(t \mid C_k)$ for every possible t, which is rather impractical for high-dimensional data sets. We view the score used in the CAEP as a (not very rigorous) "surrogate" for $P(C_k) \times P(t \mid C_k)$.

The CAEP method is the first application of emerging patterns to classification. There are several variations around the main ideas of the CAEP method. We close this section by mentioning two of these variations: the JEP method (Li et al. 2000b) and the DeEP method (Li et al. 2000a). The JEP method uses exclusively emerging patterns whose supports increases from zero in one class to nonzero in the other class; such emerging patterns are called "jumping" emerging patterns. In the situation where there are many jumping emerging patterns, the JEP method would perform well and the classifier would be more efficient to train. The DeEP method is an instance-based method in the sense that it does not have a single training phase. Instead, for each instance to be predicted, a separate training phase is involved that is automatically "customized" for that instance. So although the DeEP method has to train as many times as there are instances to be predicted, each training phase takes much less time than the CAEP method. However, as each prediction must be preceded by its corresponding training phase, the overall prediction time for the DeEP

method is typically an order of magnitude longer than the CAEP method. In terms of accuracy, the DeEP method is usually slightly better than the CAEP method. However, its biggest advantage is that it can incorporate new training data easily. This feature makes it very useful for applications where the training data must be frequently updated.

13.5 Conclusion

In this chapter, we began with a description of datamining in general and then focused on the classification and prediction aspect of datamining. Classification is the process of finding a set of models that describe and distinguish data classes or concepts. The model can then be used for the purpose of predicting the class of newly encountered data. The derived model could be in various forms such as if-then rules, decision trees, mathematical formulae, artificial neural networks, or emerging patterns.

In the MHC-binding peptide example, we encountered artificial neural networks. We gave a brief introduction to feed-forward artificial neural networks and showed how to do back propagation on these neural networks. Using the work of Brusic and Zeleznikow (1999), we demonstrated that such neural networks can be trained to differentiate peptides that bind the class-II MHC molecule, HLA-DR4, from those that do not bind this molecule. We also briefly mentioned other MHC-binding peptide classification methods (Hammer et al. 1994; Rammensee et al. 1995; Rognan et al. 1994).

In the diabetes example, we encountered emerging patterns. We gave a brief introduction to the concept of emerging patterns and showed how to perform classification by aggregating emerging patterns. Using the work of Dong et al. (1999), we demonstrated that this method can be used to differentiate diabetic Pima Indians from non-diabetic ones. We also briefly mentioned other emerging pattern-based datamining methods (Dong and Li 1999; Li et al. 2000a, b).

In addition to artificial neural networks and emerging patterns, we also briefly surveyed other techniques such as Bayesian classifiers, hidden Markov models, and support vector machines. All of these techniques have proven useful in applications such as disease diagnosis, gene finding, protein family classification, gene expression data classification, protein secondary structure prediction, and so on. However, we did not go into their details due to space constraint.

All these techniques are general knowledge discovery methods. With appropriate data preparation, they are applicable in a large variety of classification applications. However, in some situations, more specialized algorithms can produce better results, such as in the detection of translation initiation sites in RNA sequences (Hannenhalli

et al. 1999) and the detection of significant differentially expressed genes in gene expression data (Slonim et al. 2000).

Finally, we should mention that classification may need to be preceded by relevance analysis, which attempts to identify attributes that do not contribute to the classification process. These attributes can then be excluded. In our two examples, we did not use a separate relevance analysis. However, in order to deal with datasets of much higher dimension, a separate relevance analysis step is often crucial. The reason is that datamining algorithms are in general exponential in complexity with respect to the number of attributes present in the datasets. For example, a gene expression record typically contains several thousand attributes, each attribute corresponding to the expression level of a distinct gene. A fast and cheap relevance analysis should be used to reduce these thousands of genes into several tens of most relevant genes for subsequent classification analysis (Slonim et al. 2000). There are of course applications involving a large number of dimensions where a separate relevance analysis step is not necessary. For example, Temkin et al. (1995) reported that a methodology like CART (Breiman et al. 1984), which does not require a separate relevance analysis step, is able to produce a classifier of sufficiently high accuracy for predicting neurobehavioral outcome of head-injury survivors. Note also that although relevance analysis is useful in ranking the importance of each dimension of a dataset, it is not always desirable to aggressively eliminate lower ranking dimensions. For example, Spellman et al. (1998) would not have identified eight hundred cell cycle genes if they blindly reduced all the yeast genes to just the several tens of the most relevant ones!

Acknowledgments

Section 13.3 on MHC-binding peptides is based on the work by Vladimir Brusic of my lab. Section 13.4 on diabetes clinical data is based on my collaboration with Guozhu Dong, Jinyan Li, and Jenny Zhang. My lab is funded in part by the Singapore National Science and Technology Board. I am also grateful to the useful comments by the referees.

References

Adriaans, P., and Zantinge, D. (1996). *Data Mining*. Harlow, UK: Addison Wesley Longman.

Arnold, G. E., Dunker, A. K., Johns, S. L., and Douthart, R. J. (1992). Use of conditional probabilities for determining relationships between amino acid sequence and protein secondary structure. *Proteins* 12(4): 382–399.

Baldi, P., Brunak, S., Chauvin, Y., and Krogh, A. (1997). Hidden Markov models for human genes: Periodic patterns in exon sequences. In *Theoretical and Computational Methods in Genome Research*, Suhai, S., ed., 15–32, New York: Plenum.

Baldi, P., and Chauvin, Y. (1994). Hidden Markov models of G-protein-coupled receptor family. *J. Comput. Biol.* 1: 311–335.

Baldi, P., and Brunak, S. (1999). *Bioinformatics: The Machine Learning Approach*. Cambridge, Mass.: MIT Press.

Bateman, A., Birney, E., Durbin, R., Eddy, S. R., Finn, R. D., and Sonnhammer, E. L. L. (1999). Pfam 3.1: 1313 multiple alignments and profile HMMs match the majority of proteins. *Nucleic Acid. Res.* 27(1): 260–262.

Ben-Dor, A., Shamir, R., and Yakhini, Z. (1999). Clustering gene expression patterns. *J. Comput. Biol.* 6: 281–297.

Borodovsky, M., and McIninch, J. D. (1993). GENEMARK: Parallel gene recognition for both DNA strands. *Comput. Chem.* 17(2): 123–133.

Borodovsky, M., McIninch, J. D., Koonin, E. V., Rudd, K. E., Medigue, C., and Danchin, A. (1995). Detection of new genes in a bacterial genome using Markov models for three gene classes. *Nucleic Acid. Res.* 23: 3554–3562.

Borras-Cuesta, F., Golvano, J. J., Garcia-Granero, M., Sarobe, P., Riezu-Boj, J. I., Huarte, E., and Lasarte, J. J. (2000). Specific and general HLA-DR binding motifs: Comparison of algorithms. *Human Immunol.* 61(3): 266 278.

Breiman, L., Friedman, L., Olshen, R., and Stone, C. (1984). *Classification and Regression Trees*. Belmont, Calif.: Wadsworth.

Brown, M. P., Grundy, W. N., Lin, D., Cristianini, N., Sugnet, C. W., Furey, T. S., Ares, M. Jr., and Haussler, D. (2000). Knowledge-based analysis of microarray gene expression data by using support vector machines. *Proc. Natl. Acad. Sci. USA* 97(1): 262–267.

Brusic, V., Rudy, G., and Harrison, L. C. (1994). Prediction of MHC binding peptides using artificial neural network. In *Complex Systems: Mechanism of Adaptation*, Stonier, R. J., and Yu, X. H., eds., 253–260. Washington, DC: IOS Press.

Brusic, V., Rudy, G., and Harrison, L. C. (1998a). MHCPEP, a database of MHC-binding peptides: Update 1997. *Nucleic Acid. Res.* 26: 368–371.

Brusic, V., Rudy, G., Honeyman, M. C., and Harrison, L. C. (1998b). Prediction of MHC class II binding peptides using an evolutionary algorithm and artificial neural network. *Bioinformatics* 14(2): 121 130.

Brusic, V., and Zeleznikow, J. (1999). Computational binding assays of antigenic peptides. *Lett. Pept. Sci.* 6: 313–324.

Burden, F. R., and Winkler, D. A. (2000). A quantitative structure-activity relationships model for the acute toxicity of substituted benzenes to *tetrahymena pyriformis* using Bayesian-regularised neural networks. *Chem. Res. Toxicol.* 13: 436–440.

Burges, C. J. C. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining Knowl. Dis.* 2(2): 121–167.

Chauvin, Y., and Rumelhart, D. (1995). *Backpropagation: Theory, Architectures, and Applications*. Hillsdale, N.J.: Lawrence Erlbaum.

Claros, M. G., Brunak, S., and von Heijne, G. (1997). Prediction of n-terminal protein sorting signals. *Curr. Opin. Struct. Biol.* 7: 394–398.

Demsar, J., Zupan, B., Kattan, M. W., Beck, J. R., and Bratko, I. (1999). Naive Bayesian-based nomogram for prediction of prostate cancer recurrence. *Stud. Health Technol. Inform.* 68: 436–441.

D'haeseleer, P., Wen, X., Fuhrman, S., and Somogyi, R. (1998). Mining the gene expression matrix: Inferring gene relationships from large scale gene expression data. In *Information Processing in Cells and Tissues*, Paton, R. C., and Holcombe, M., eds., 203–212. New York: Plenum.

Dong, G., and Li, J. (1999). Efficient mining of emerging patterns: Discovering trends and differences. In *Proc. 5th ACM SIGKDD Intl. Conf. on Knowledge Discovery & Data Mining*, 15 18. New York: ACM Press.

Dong, G., Zhang, X., Wong, L., and Li, J. (1999). CAEP: Classification by aggregating emerging patterns. In *LNCS 1721: Discovery Science*, Arikawa, S., and Furukawa, K., eds., 30–42. Berlin: Springer.

Duda, R., and Hart, P. (1973). *Pattern Classification and Scene Analysis*. New York: Wiley.

Durbin, R., Eddy, S. R., Krogh, A., and Mitchison, G. (1998). *Biological Sequence Analysis: Probabilistics Models of Proteins and Nucleic Acids*. Cambridge, UK: Cambridge University Press.

Eddy, S. R. (1996). Hidden Markov models. *Curr. Opin. Struct. Biol.* 6: 361–365.

Eisen, M. B., Spellman, P. T., Brown, P. O., and Bostein, D. (1998). Cluster analysis and display of genome-wide expression patterns. *Proc. Natl. Acad. Sci. USA* 95: 14863–14868.

Emanuelsson, O., Nielsen, H., and von Heijne, G. (1999). ChloroP, a neural network-based method for predicting chloroplast transit peptides and their cleavage sites. *Protein Sci.* 8(5): 978–984.

Di Francesco, V., Granier, J., and Munson, P. J. (1997). Protein topology recognition from secondary structure sequences—applications of the hidden Markov models to the alpha class proteins. *J. Mol. Biol.* 267: 446–463.

Friess, T. T., Cristianini, N., and Campbell, C. (1998). The kernel adatron algorithm: A fast and simple learning procedure for support vector machines. In *Proc. 15th Intl. Conf. on Machine Learning*, Shavlik, J., ed. San Francisco: Morgan Kaufmann.

Gehrke, J., Ganti, V., Ramakrishnan, R., and Loh, W. Y. (1999). BOAT—optimistic decision tree construction. In *Proc. ACM-SIGMOD Intl. Confl. on Management of Data*, Delis, A., Faloutsos, C., and Ghandeharizadeh, S., eds., 169–180. New York: ACM Press.

Ghosh, S., and Majumder, P. P. (2000). Mapping a quantitative trait locus via the EM algorithm and Bayesian classification. *Genet. Epidemiol.* 19(2): 97–126.

Gulukota, K. (1998). Skipping a step with neural nets. *Nature Biotechnol.* 16(8): 722–723.

Hammer, J., Nagy, Z. A., and Sinigaglia, F. (1994). Rules governing peptide-class II MHC molecule interactions. *Behring Inst. Mitt.* 94: 124–132.

Han, J., and Kamber, M. (2000). *Data Mining: Concepts and Techniques*. San Francisco: Morgan Kaufmann.

Hannenhalli, S. S., Hayes, W. S., Hatzigeorgiou, A. G., and Fickett, J. W. (1999). Bacterial start site prediction. *Nucl. Acid. Res.* 27(17): 3577–3582.

Heckerman, D. (1996). Bayesian networks for knowledge discovery. In *Advances in Knowledge Discovery and Data Mining*, Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P., and Uthurusamy, R., eds., 273–305. Cambridge, Mass.: MIT Press.

Hennessy, D., Buchanan, B., Subramanian, D., Wilkosz, P. A., and Rosenberg, J. M. (2000). Statistical methods for the objective design of screening procedures for macromolecular crystallization. *Acta Crystallogr. D Biol. Crystallogr.* 56(7): 817–827.

Honeyman, M. C., Brusic, V., Stone, N., and Harrison, L. C. (1998). Neural network-based prediction of candidate T-cell epitopes. *Nature Biotech.* 16(10): 966–969.

Huitema, A. D., Mathot, R. A., Tibben, M. M., Schellens, J. H., Rodenhuis, S., and Beijnen, J. H. (2000). Validation of techniques for the prediction of carboplatin exposure: Application of Bayesian methods. *Clin. Pharmacol. Ther.* 67(6): 621–630.

Jaakkola, T., Diekhans, M., and Haussler, D. (2000). A discriminative framework for detecting remote protein homologies. *J. Comput. Biol.* 7(1–2): 95–114.

Jensen, F. V. (1996). *An Introduction to Bayesian Networks*. Berlin: Springer-Verlag.

John, G. H. (1997). *Enhancements to the Data Mining Process*. PhD thesis, Stanford University.

Kaas, G. V. (1980). An exploratory technique for investigating large quantities of categorical data. *Appl. Stat.* 29: 119–127.

Kasif, S., and Delcher, A. L. (1998). Modeling biological data and structure with probabilistic networks. In *Computational Methods in Molecular Biology*, Salzberg, S. L., Searls, D. B., and Kasif, S., eds., 335–352. Amsterdam: Elsevier.

Kohavi, R., and Sahami, M. (1996). Error-based and entropy-based discretization of continuous features. In *Proc. 2nd Intl. Conf. on Knowledge Discovery and Data Mining*, Simoudis, E., Han, J., and Fayyad, U. M., eds., 114–119. Menlo Park, Calif.: AAAI Press.

Krogh, A. (1998). An introduction to hidden Markov models for biological sequences. In *Computational Methods in Molecular Biology*, Salzberg, S. L., Searls, D. B., and Kasif, S., eds., 45–62. Amsterdam: Elsevier.

Krogh, A., Brown, M., Mian, I. S., Sjolander, K., and Haussler, D. (1994). Hidden Markov models in computational biology: Applications to protein modeling. *J. Mol. Biol.* 235: 1501–1531.

Kukar, M., Kononenko, I., and Silvester, T. (1996). Machine learning in prognosis of the femoral neck fracture recovery. *Artif. Intell. Med.* 8(5): 431–451.

Lauritzen, S. L. (1995). The EM algorithm for graphical association models with missing data. *Comput. Stat. Data Anal.* 19: 191–201.

Li, J., Dong, G., and Ramamohanarao, K. (2000a). DeEPs: Instance-based classification using emerging patterns. In *LNCS 1910: Proc. 4th Eur. Conf. on Principles and Practice of Knowledge Discovery in Databases*, Zighel, D. A., Komorowski, H. J., and Zytkow, J. M., eds., 191–200. Berlin: Springer.

Li, J., Dong, G., and Ramamohanarao, K. (2000b). Making use of the most expressive jumping emerging patterns for classification. In *LNCS 1805: Proc. 4th Pacific-Asia Conf. on Knowledge Discovery in Databases*, Terano, T., Liu, H., and Chen, A. L. P., eds., 220–232. Berlin: Springer.

Lincoff, G. H. (1981). *The Audubon Society Field Guide to North American Mushrooms*. New York: Knopf.

Liu, B., Hsu, W., and Ma, Y. (1998). Integrating classification and association rule mining. In *Proc. 4th Intl. Confl. on Knowledge Discovery in Databases and Data Mining*, Agrawal, R., Stolorz, P. E., and Piatesky-Shapiro, G., eds., 80–86. Menlo Park, Calif.: AAAI Press.

Loh, W. Y., and Shih, Y. S. (1997). Split selection methods for classification trees. *Statist. Sinica* 7: 815–840.

Loh, W. Y., and Vanichsetakul, N. (1988). Tree-structured classification via generalized discriminant analysis. *J. Am. Stat. Assoc.* 83: 715–728.

Longacre, T. A., Chung, M. H., Jensen, D. N., and Hendrickson, M. R. (1995). Proposed criteria for the diagnosis of well-differentiated endometrial carcinoma. A diagnostic test for myoinvasion. *Am. J. Surg. Pathol.* 19(4): 371–406.

Lossos, I. S., Breuer, R., Intrator, O., and Lossos, A. (2000). Cerebrospinal fluid lactate dehydrogenase isoenzyme analysis for the diagnosis of central nervous system involvement in hematooncologic patients. *Cancer* 88(7): 1599–1604.

Lowe, T. M., and Eddy, S. R. (1997). tRNAscan-SE: A program for improved detection of transfer RNA genes in genomic sequence. *Nucl. Acid. Res.* 25(5): 955–964.

Lowe, T. M., and Eddy, S. R. (1999). A computational screen for methylation guide snoRNAs in yeast. *Science* 283: 1168–1171.

Mangasarian, O. L., Street, W. N., and Wolberg, W. H. (1995). Breast cancer diagnosis and prognosis via linear programming. *Oper. Res.* 43(4): 570–577.

McCue, L. A., McDonough, K. A., and Lawrence, C. E. (2000). Functional classification of cnmp-binding proteins and nucleotide cyclases with implications for novel regulatory pathways in mycobacterium tuberculosis. *Genome Res.* 10(2): 204–219.

Mehta, M., Agrawal, R., and Rissanen, J. (1996). SLIQ: A fast scalable classifier for data mining. In *LNCS 1057: Proc. Intl. Confl. on Extending Database Technology*, Apers, P. M. G., Bouzeghoub, M., and Gardarin, G., eds., 18–32. Berlin: Springer.

Mitchell, T. M. (1997). *Machine Learning*. New York: McGraw-Hill.

Miyata, Y. (1991). *A User's Guide to PlaNet Version 5.6*. University of Colorado.

Nielsen, H., Engelbrecht, J., Brunak, S., and von Heijne, G. (1994). Identification of prokaryotic and eukaryotic signal peptides and prediction of their cleavage sites. *Protein Sci.* 3: 3–14.

Patterson, D. (1996). *Artificial Neural Networks: Theory and Applications*. Singapore: Prentice Hall.

Pedersen, A. G., Baldi, P., Brunak, S., and Chauvin, Y. (1996). Characterization of prokaryotic and eukaryotic promoters using hidden Markov models. *Intell. Syst. Mol. Biol.* 4: 182–191.

Pedersen, A. G., and Nielsen, H. (1997). Neural network prediction of translation initiation sites in eukaryotes: Perspectives for EST and genome analysis. *Intell. Syst. Mol. Biol.* 5: 226–233.

Qian, N., and Sejnowski, T. J. (1988). Predicting the secondary structure of globular proteins using neural network models. *J. Mol. Biol.* 202: 865–884.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning* 1: 81–106.

Quinlan, J. R. (1992). *C4.5: Programs for Machine Learning*. San Mateo, Calif.: Morgan Kaufmann.

Raddrizzani, L., and Hammer, J. (2000). Epitope scanning using virtual matrix-based algorithms. *Brief. Bioinformat.* 1(2): 179–189.

Rainer, T. H., et al. (1999). Derivation of a prediction rule for post-traumatic acute lung injury. *Resuscitation* 42(3): 187–196.

Rammensee, H. G., Friede, T., and Stevanovlic, S. (1995). MHC ligands and peptide motifs: First listing. *Immunogenetics* 41(4): 178–228.

Rastogi, R., and Shim, K. (1998). Public: A decision tree classifier that integrates building and pruning. In *Proc. 24th Intl. Conf. Very Large Data Bases*, Gupta, A., Shmueli, O., and Widom, J., eds., 404–415. San Francisco: Morgan Kaufmann.

Raudys, S. (2000). How good are support vector machines? *Neural Network* 13(1): 17–19.

Riis, S. K., and Krogh, A. (1996). Improving prediction of protein secondary structure using structured neural networks and multiple sequence alignments. *J. Comput. Biol.* 3: 163–183.

Rognan, D., Scapozza, L., Folkers, G., and Daser, A. (1994). Molecular dynamics simulation of MHC-peptide complexes as a tool for predicting potential T cell epitopes. *Biochemistry* 33(38): 11476–11485.

Rost, B., and Sander, C. (1994). Combining evolutionary information and neural networks to predict protein secondary structure. *Proteins* 19: 55–72.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by backpropagating errors. *Nature* 323: 533–536.

Russell, S., Binder, J., Koller, D., and Kanazawa, K. (1995). Local learning in probabilistic networks with hidden variables. In *Proc. 14th Joint Intl. Conf. Artificial Intelligence*, vol. 2, 1146–1152. San Francisco: Morgan Kaufmann.

Salzberg, S. L., Delcher, A. L., Kasif, S., and White, O. (1998). Microbial gene identification using interpolated Markov models. *Nucl. Acid. Res.* 26(2): 544–548.

Savoie, C. J., Kamikawaji, N., and Sasazuki, T. (1999). Use of BONSAI decision trees for the identification of potential MHC Class I peptide epitope motifs. In *Proc. Pacific Symp. Biocomput.*, Altman, R. B., Dunker, A. K., Hunter, L., Klein, T. E., and Lauderdale, K., eds., 182–189. Singapore: World Scientific.

Schapire, R. E. (1990). The strength of weak learnability. *Machine Learning* 5(2): 197–227.

Schoenbach, C., Koh, J., Sheng, X., Wong, L., and Brusic, V. (2000a). FIMM, a database of functional molecular immunology. *Nucl. Acid. Res.* 28(1): 222–224.

Schoenbach, C., Kowalski-Saunders, P., and Brusic, V. (2000b). Data warehousing in molecular biology. *Brief. Bioinformat.* 1: 190–198.

Scott, J. A., Palmer, E. L., and Fischman, A. J. (2000). How well can radiologists using neural network software diagnose pulmonary embolism? *Am. J. Roentgenol.* 175(2): 399–405.

Selker, H. P., Griffith, J. L., Patil, S., Long, W. J., and D'Agostino, R. B. (1995). A comparison of performance of mathematical predictive methods for medical diagnosis: identifying acute cardiac ischemia among emergency department patients. *J. Invest. Med.* 43(5): 468–476.

Shafer, J., Agrawal, R., and Mehta, M. (1996). SPRINT: A scalable parallel classifier for data mining. In *Proc. 22nd Intl. Conf. Very Large Data Bases*, Vijayaraman, T. M., Buchmann, A. P., Mohan, C., and Sarda, N. L., eds., 544–555. San Francisco: Morgan Kaufmann.

Slonim, D. K., Tamayo, P., Mesirov, J. P., Golub, T. R., and Lander, E. S. (2000). Class prediction and discovery using gene expression data. In *Proc. 4th Intl. Conf. Computational Molecular Biology*, 262–271. New York: ACM Press.

Smith, J. W., Everhart, J. E., Dickson, W. C., Knowler, W. C., and Johannes, R. S. (1988). Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. In *Proc. Symp. Computer Applications and Medical Care*, 261–265. IEEE Computer Society Press.

Snyder, E. E., and Stormo, G. D. (1995). Identification of protein coding regions in genomic DNA. *J. Mol. Biol.* 248: 1–18.

Spellman, P. T., Sherlock, G., Zhang, M. Q., Iyer, V. R., Anders, K., Eisen, M. B., Brown, P. O., Botstein, D., and Futcher, B. (1998). Comprehensive identification of cell cycle-regulated genes of the yeast saccharomyces cerevisiae by microarray hybridization. *Mol. Biol. Cell.* 9(12): 3273–3297.

Steeg, E. W. (1993). Neural networks, adaptive optimization, and RNA secondary structure prediction. In *Artificial Intelligence and Molecular Biology*, Hunter, L., ed., 121–160. Menlo Park, Calif.: AAAI Press.

Stryer, L. (1995). *Biochemistry*. New York: Freeman.

Stultz, C. M., Nambudripad, R., Lathrop, R. H., and White, J. V. (1997). Predicting protein structure with probabilistic models. In *Protein Structural Biology in Biomedical Research*, Allewell, N., and Woodward, C., eds., 447–506. Greenwich, CT: JAI Press.

Temkin, N. R., Holubkov, R., Machamer, J. E., Winn, H. R., and Dikmen, S. S. (1995). Classification and regression trees (CART) for prediction of function at 1 year following head trauma. *J. Neurosurg.* 85(5): 764–771.

Turton, E. P., Scott, D. J., Delbridge, M., Snowden, S., and Kester, R. C. (2000). Ruptured abdominal aortic aneurysm: A novel method of outcome prediction using neural network technology. *Eur. J. Vasc. Endovasc. Surg.* 19(2): 184–189.

Uberbacher, E. C., and Mural, R. J. (1991). Locating protein-coding regions in human DNA sequences by a multiple sensor-neural network approach. *Proc. Natl. Acad. Sci. USA* 88: 11261–11265.

Utgoff, P. E. (1988). ID5: An incremental ID3. In *Proc. 5th Intl. Conf. Machine Learning*, Laird, J. E., ed., 107–120. San Francisco: Morgan Kaufmann.

Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory*. Berlin: Springer.

Vriesema, J. L., van der Poel, H. G., Debruyne, F. M., Schalken, J. A., Kok, L. P., and Boon, M. E. (2000). Neural network-based digitized cell image diagnosis of bladder wash cytology. *Diagn. Cytopathol.* 23(3): 171–179.

Yada, T., Ishikawa, M., Tanaka, H., and Asai, K. (1996). Extraction of hidden Markov model representations of signal patterns in DNA sequences. In *Proc. Pacific Symp. Biocomput.*, 686–696. Singapore: World Scientific.

Zien, A., Raatsch, G., Mika, S., Schoelkopf, B., Lemmem, C., Smola, A., Lengauer, T., and Mueller, K. R. (1999). Engineering support vector machine kernels that recognize translation initiation sites. In *German Conf. Bioinformatics*, http://www.bioinfo.de/isb/gcb99/talks/zien.

IV COMPUTATIONAL STRUCTURAL BIOLOGY

14 RNA Secondary Structure Prediction

Zhuozhi Wang and Kaizhong Zhang

14.1 Introduction

Ribonucleic Acid (RNA) is an important molecule that performs a wide range of functions in biological systems. In particular, it is RNA (not DNA) that contains the genetic information of viruses such as HIV and thereby regulates the functions of these viruses. RNA has recently become the center of much attention because of its catalytic properties (Cech and Bass 1988), leading to an increased interest in obtaining structural information.

RNA molecules have two sets of structural information: first, the *primary structure* of RNA is a single strand made of the ribonucleotides A (adenine), C (cytosine), G (guanine), and U (uracil). Secondly, the ribonucleotide sequences fold over onto themselves to form double-stranded regions of base pairings, yielding higher order *tertiary structures.*

It is well known that the structural features of RNAs are important in the molecular mechanisms involving their functions. The presumption, of course, is that to a preserved function there corresponds a preserved molecular confirmation and, therefore, a preserved structure. The RNA *secondary structure* is a restricted subset of the tertiary structure that plays an important role between primary structure and tertiary structure, as the problem of inferring the tertiary structures of RNA molecules is often intractable. Based on a reliable secondary structure, the possible tertiary interactions that occur between secondary structural elements, as well as between these elements and single-stranded regions of RNAs, can be characterized.

Currently, the only completely accurate method of determining the folded structure of an RNA molecule is by X-ray crystallography; however, this is not only time consuming, but also expensive.

The use of computational methods to predict RNA secondary structure began more than 30 years ago. Many computational methods have been proposed in an attempt to predict RNA secondary structures. Although computational methods sometimes only provide an approximate RNA structural model, they facilitate the future study of RNA structures. To date, several approaches have been established for predicting RNA secondary structure, most notably the phylogenetic comparative method (James et al. 1989; Winker et al. 1990; Chiu and Kolodziejczak 1991; Chan et al. 1991; Gutell et al. 1992; Eddy and Durbin 1994; Gutell et al. 1994; Le et al. 1995), the thermodynamic energy minimization method (Nussinov et al. 1978; Waterman 1978; Zuker and Stiegler 1981; Turner et al. 1988; Zuker 1989; Rivas and Eddy 1999; Zuker 2000), and the stochastic context-free grammar method (Searls

1992, 1993; Eddy and Durbin 1994; Sakakibara et al. 1994; Knudsen and Hein 1999). There are other approaches for RNA secondary structure prediction, including the equilibrium partition function method (McCaskill 1990), genetic algorithms (Shapiro and Wu 1996), algorithms that combine the phylogenetic and the thermodynamic methods (Sankoff 1985; Le and Zuker 1990; Wang and Zhang 1999), and method based on the hybridization of the thermodynamic and the phylogenetic methods as well as genetic algorithm (Chen et al. 2000). In this chapter we discuss algorithms/methods to predict RNA secondary structure.

14.2 Basic Definitions

DEFINITION 1 (Primary Structure) Because an RNA sequence is composed of four possible bases, we can use a four-letter alphabet to represent an RNA sequence, $\Sigma = \{A, C, G, U\}$. This base sequence is usually referred to as *primary structure*. Formally, it is written as follows:

$$R = r_1, r_2, \ldots, r_n, \qquad r_i \in \Sigma$$

Following convention, we denote the left end of the sequence as the $5'$ end and the right side of the sequence as the $3'$ end.

An RNA sequence folds by intramolecular base pairing and is stabilized by the hydrogen bonds that result from that base pairing. Additionally, the stacking of base pairs in a helix stabilizes the molecule and decreases the free energy of the folded structure, but the appearance of loops (see the definition below) destabilizes the molecule and increases the free energy of the RNA structure.

DEFINITION 2 (Canonical Base Pairs) In an RNA secondary structure, base pairs are formed as one of the three kinds of pairs, C-G (G-C), A-U (U-A), and G-U (U-G). There are three hydrogen bonds between C-G (G-C) pairs, two between A-U (U-A), and one between G-U (U-G). Base pairs C-G (G-C) and A-U (U-A) are called Watson-Crick base pairs. The base pair G-U (U-G) is referred to as a wobble base pair. These three types of pairings are referred to as *canonical base pairs*.

DEFINITION 3 (Secondary Structure) We use $i \cdot j$ to represent the base pair formed by the ith base, r_i, and the jth base, r_j, where $1 \le i < j \le n$. Let S be a set of base pairs for sequence R, then set S is called *RNA secondary structure* if S satisfies the following conditions:

1. For any base pair $i \cdot j$ in S, the base pairing of r_i and r_j is a *canonical base pair* (see definition 2);

2. For any two base pairs $i_1 \cdot j_1$ and $i_2 \cdot j_2$, either $i_1 = i_2$ and $j_1 = j_2$ or $i_1 \neq i_2$, $i_1 \neq j_2$, $j_1 \neq i_2$ and $j_1 \neq j_2$;
3. If $h < i < j < k$, then S cannot contain both $r_h \cdot r_j$ and $r_i \cdot r_k$;
4. If S contains $r_i \cdot r_j$, then $|j - i| \geq 4$.

Following condition 2, a base r_i can be in one of the two states: either it is paired with another base, or it is unpaired. Condition 3 is usually referred to as the non-crossing condition. In reality, conditions 2 and 3 may not be true, which will result in "triples," "knots," and so on; however, all these are considered as features of the higher level structure (that is, the *tertiary structure*). Still, it is possible to violate condition 1, which will result in noncanonical pairs, such as G·A.

Obviously, the RNA secondary structure is much more complicated than the RNA primary structure.

To facilitate the study of RNA secondary structure, the RNA secondary structures are analytically decomposed, using a process called K-loop decomposition, into five kinds of substructures, namely *stacked pairs, hairpin loops, bulge loops, interior loops*, and *multiple loops* (sometimes we group the latter four simply as *loops*).

DEFINITION 4 (K-loop Decomposition) If $i \cdot j$ is a base pair and $i < k < j$, we say that k is accessible from $i \cdot j$ if there is no $i' \cdot j'$ such that $i < i' < k < j' < j$. Similarly, we say that the base pair $k \cdot l$ is accessible from $i \cdot j$ if both k and l are accessible from $i \cdot j$. The set of $(k - 1)$ base pairs and k' single-stranded bases accessible from $i \cdot j$ is called the k-loop closed by $i \cdot j$. The null k-loop, s_0, consists of those single bases and base pairs accessible from no base pair. They are also called the free bases and base pairs. Any secondary structure S decomposes the set $\{1, 2, \ldots, n\}$ uniquely into k-loops $s_0, s_1, \ldots, s_m$, where $m > 0$ if and only if $S \neq \varnothing$. This decomposition was first introduced by Sankoff et al. (1983). The present definition follows Zuker and Sankoff (1984) and Zuker (1986), where the closing base pair is not contained in the k-loop. A k-loop is sometimes also referred to as a k-cycle.

Biochemists had developed their nomenclature for k-loops long before any formal definition was given. The various cases and subcases are as follows:

- $k = 1$: A 1-loop is called a hairpin loop.
- $k = 2$: Let $i' \cdot j'$ be the base pair accessible from $i \cdot j$. Then the 2-loop is called a
  1. stacked pair if $i' - i = 1$ and $j - j' = 1$, a
  2. bulge loop if $i' - i > 1$ or $j - j' > 1$, but not both, and an
  3. interior loop if both $i' - i > 1$ and $j - j' > 1$.
- $k > 2$: These k-loops are called multi-branched or multiple loops.

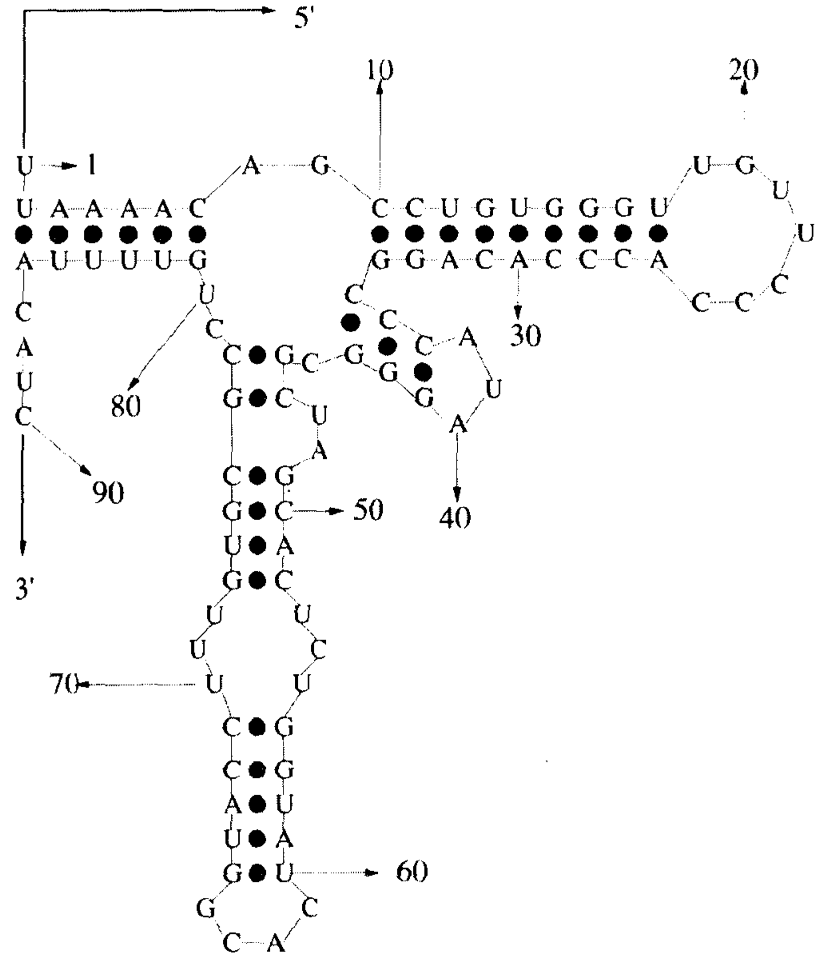


Figure 14.1
An RNA secondary structure.

Figure 14.1 shows an RNA secondary structure. The nucleotides are laid out in such a way that paired bases are proximal with a • indicating the base pairing.

• In figure 14.1, base pair 18 · 26 encloses a *hairpin loop*; all the bases that are surrounded by base pair 18 · 26 are not paired.

• Base pairs 2 · 86, 3 · 85, 4 · 84, 5 · 83, and 6 · 82 as a whole is called a *helix* or *stem*. Generally, only more than two consecutive pairs will be referred to as a helix or stem.

• The area closed by base pair 52 · 73 and 56 · 69 is called an *interior loop*. Note that all the bases are single bases.

• Base pairs 46 · 77 and 49 · 76 enclose a *bulge loop*; all the bases between 46 and 49 are not base paired.

• In this figure, a *multiple loop* is formed by the base pair $7 \cdot 81$, single bases 8–9, base pairs $10 \cdot 34$ and $35 \cdot 43$, single base 44, base pair $45 \cdot 78$, and single bases 79–80.

• The *(external) single-stranded region* consists of the single bases that are not surrounded by any base pairs in the structure, in this case single base 1 and single bases 87–90.

14.3 Combinatorial Algorithm

In 1975, Pipas and McMahon published the *combinatorial method*, which is believed to be the first widely used method of predicting RNA secondary structures by using the free energy rules. The main idea of this algorithm is to form the structures by combining all potential helices in all possible ways. This algorithm works well when dealing with short RNA sequences, but is infeasible for long sequences. The algorithm consists of three steps.

In the first step, the RNA sequence is read and a bonding matrix B and a compatible matrix C are constructed.

A complete bonding matrix B for the sequence is set up by using the following rules:

• If base i is able to form a classical Watson-Crick hydrogen-bonding pair, C·G, G·C, A·U, or U·A, with base j, then the matrix element $B_{i,j} = 1$.

• If base i forms a G·U or U·G pair with base j, then $B_{i,j} = 2$.

• If base i and base j do not form the kind of base pairs discussed above, $B_{i,j} = 0$.

From this matrix, all stable helical regions can be determined. A stable helical region is defined as three or more consecutive base pairs ordered such that the strands are antiparallel. On the bonding matrix this corresponds to find a set, $\{B_{i,j}, B_{i+1,j-1}, \ldots, B_{i+m,j-m}\}$, where all elements are nonzero.

After the program has compiled the list of all possible helical regions derivable from the given primary sequence, it will set up a compatibility matrix (C-matrix) to indicate whether two regions can occur together in a given secondary structure. The elements of this matrix are defined as:

• Let i, j be two helical regions, if i and j are compatible, which means that these two helical regions can exist together in a given structure, then the matrix element will be $C_{i,j} = 1$, otherwise, $C_{i,j} = 0$.

The following two criteria are given in the paper. The first criterion excludes overlapping helices. The second criterion disallows pseudo-knots.

• Let i and j be two helical regions and let R_i and R_j be two sets whose elements are the bases forming region i and j respectively. Then $C_{i,j}$ will be assigned 1 if and only if $R_i \cap R_j = \emptyset$; otherwise $C_{i,j}$ will be given 0. This is used to avoid conflicting stems.

• Let i and j be two helical regions and let H_i be a set whose elements are the bases enclosed by region i. Let Q_i be a set whose elements are all bases not included in R_i and H_i. Then $C_{i,j}$ will be assigned 1 if and only if $R_j \cap H_i = \emptyset$ or $R_j \cap Q_i = \emptyset$; otherwise $C_{i,j} = 0$.

The second step is permutation. The main purpose of this step is to create all possible structures that are obtainable from the given polynucleotide sequence. This is accomplished by generating all possible permutations of the nonzero (compatible) elements in the C-matrix. Here, a structure is defined as a set of three or more compatible helical regions.

In the third step, all the generated structures are evaluated, base by base, and assigned a total free energy. The structures are then ordered by their free energies, and the best (e.g., the most negative one) will be selected out and considered as the optimal structure from the given RNA primary structure. Favorable free energy contributions are assumed to be made by stacking interactions of stacked base pairs. The specific values assigned are empirical. Unfavorable free energy contributions are assigned to loops, as originally proposed by Tinoco et al. (1971).

This method can easily handle relatively short sequences, such as transfer RNAs. However, it is not very efficient for folding long RNA sequences: the time needed for long RNA sequences is at least proportional to 2^n, where n is the number of nucleotides in the sequence. Its major advantage is that it can predict a great many different RNA secondary structures.

14.4 Energy Minimization Algorithms

The main idea of free energy minimization algorithms is that the whole structure can be considered as a collection of substructures; thus, if one can obtain all the optimal substructures, the whole optimal structure can also be determined. Generally, an energy minimization method has two stages. One is the *filling algorithm* that computes the energy values of optimal structure of all fragments of a sequence. The other is the *trace back* algorithm that computes an optimal structure by searching systematically through the matrix of stored energy values.

Energy minimization algorithms all use dynamic programming. Dynamic programming methods for RNA structure began with Nussinov et al. (1978) and Waterman (1978). Since then, many improvements have been made (Zuker and Stiegler

1981; Zuker and Sankoff 1984; Waterman and Smith 1986; Zuker 1989; Lyngø et al. 1999). Currently the most popular software for RNA folding, MFOLD, is based on Zuker's algorithm (Mathews et al. 1999).

14.4.1 Base Pair Dependent Energy Minimization Algorithm

Nussinov et al. (1978) were the first to apply the dynamic programming method to the folding problem by maximizing base pairings. Nussinov's algorithm is recursive. It computes the optimal structure for small subsequences, and works its way up toward larger and larger subsequences. The recurrence relation of Nussinov's algorithm is as follows:

$$M_{i,j} = \max\left\{ M_{i+1,j}, M_{i,j-1}, M_{i+1,j-1} + 1, \max_{i<k<j}\{M_{i,k} + M_{k+1,j}\} \right\}$$

where the $M_{i,j}$ represents the maximum number of base pairs a secondary structure from subsequence $r_i, \ldots, r_j$ can possess. This equation states that the maximum number of base pairs between i and j is computed from the maximum number between $i+1$ and j, the maximum number between i and $j-1$, one (because i pairs with j) plus the maximum number between $i+1$ and $j-1$, and the maximum number between i to k and between $k+1$ to j for some k.

The above equation is a very simple one, in which only the number of base pairs is considered, and the biological/chemical information is ignored. One improvement is to assign energy value to each base pair in a secondary structure. Suppose that there is a function e such that $e(r_i, r_j)$ is the energy value of a base pair $i \cdot j$. The above equation can be modified as:

$$W_{i,j} = \min\left\{ W_{i+1,j}, W_{i,j-1}, W_{i+1,j-1} + e(r_i, r_j), \min_{i<k<j}\{W_{i,k} + W_{k+1,j}\} \right\}$$

Unfortunately, this simple method cannot fully reflect the destabilizing effects of various loops, or the nearest neighbor interactions in helices and loops.

14.4.2 Loop Dependent Energy Minimization Algorithms

An RNA molecule tends to exist in a conformation characterized by the minimum free energy E. The exact calculation of E from basic physical and chemical principles for an arbitrary secondary structure S of an RNA is not feasible. However, free energy E of an RNA secondary structure can be approximated as the sum of individual contributions from stacked pairs and loops.

The weakness of the base pair dependent energy minimization method is that it cannot easily be modified to incorporate stacking, stabilizing, and loop destabilizing energy rules.

The most sophisticated energy minimization algorithm for RNA secondary structure prediction is the Zuker algorithm. In Zuker's algorithm, the entire RNA secondary structure S is considered as a set of loops (K-loop decomposition), $s_0 \ldots s_m$, where $m > 0$. Note that stacked pairs are also referred as loops. Energies are assigned to the k-loops, and the energy of a structure S can be calculated as:

$$E(S) = \sum_{i=0}^{m} e(s_i)$$

Note that e is now a function of k-loops instead of a function of base pairs. In general, $e(s)$ is only negative when s is a stacked pair, because only stacked pairs contribute directly to the stability of the molecule. Hairpin loops, bulge loops, interior loops, and multiple loops all make positive energy contributions and therefore reduce the stability of the molecule. The goal of the algorithm is to find a secondary structure that minimizes the free energy.

Energy parameters for various loops have been fitted to the results of experimental thermodynamic studies of small model RNAs (Freier et al. 1986; Turner et al. 1987, 1988). They include parameters for stacking, hairpin loop lengths, bulge loop lengths, interior loop lengths, multiple loop lengths, single dangling bases, and terminal mismatches on stems.

For the exterior-loop, s_0, the energy is set to zero. We use $\mathrm{eh}(i, j)$ to represent the free energy of a *hairpin* loop closed by pair $i \cdot j$. We use $\mathrm{ebi}(i, j, i', j')$ to represent the free energy of the *bulge* or *interior loop* closed by the two base pairs $i \cdot j$ and $i' \cdot j'$. We use $\mathrm{es}(i, j)$ to represent the free energy of stacked pairs, $i \cdot j$ and $(i+1) \cdot (j-1)$.

A Simple Loop Dependent Energy Minimization Algorithm We first discuss a simplified version of Zuker's algorithm where the energy contribution of multiple loops is set to zero and dangling single bases have no effect.

For $i < j$, let $W(i, j)$ be the minimum folding energy of all non-empty foldings on the subsequence $r_i, \ldots, r_j$ and $V(i, j)$ be the minimum energy of all foldings on the subsequence $r_i, \ldots, r_j$ that contains the base pair $i \cdot j$. Initially, $W(i, j) = V(i, j) = +\infty$, for those $|j - i| < 4$. The recurrence equation for W is as follows:

$$W(i, j) = \min \begin{cases} W(i+1, j), & (1) \\ W(i, j-1), & (2) \\ V(i, j), & (3) \\ \min\limits_{i<k<j} \{W(i, k) + W(k+1, j)\} & (4) \end{cases}$$

This equation is obtained by considering the following cases. The first case is that, in the optimal structure, base i does not pair with any other base; therefore we have $W(i, j) = W(i+1, j)$. The second case is that, in the optimal structure, base j does not pair with any other base, and therefore we have $W(i, j) = W(i, j-1)$. The third case is that, in the optimal structure, both base i and base j are paired and they pair to each other, and therefore we have $W(i, j) = V(i, j)$. The fourth case is that, in the optimal structure, both base i and base j are paired but they do not pair to each other. Because in secondary structures there are no crossings, the optimal structure will come from two optimal substructures from subsequences $i, \ldots, k$ and $k+1, \ldots, j$ for some k. Therefore, we have

$$W(i, j) = \min_{i \le k < j} \{W(i, k) + W(k+1, j)\}$$

We now consider how to compute $V(i, j)$. Consider the optimal structure; there are several possibilities. Base pair $i \cdot j$ can be part of a hairpin, or can be part of a bulge or an interior loop, or can be stacked on base pair $(i+1) \cdot (j-1)$, or can be part of a multiple loop. This leads to the formula

$$V(i, j) = \min \begin{cases} \mathrm{eh}(i, j), & (1) \\ \mathrm{es}(i, j), & (2) \\ \mathrm{VBI}(i, j), & (3) \\ \mathrm{VM}(i, j) & (4) \end{cases}$$

where $\mathrm{VBI}(i, j)$ is for bulge or interior loop and $\mathrm{VM}(i, j)$ is for multiple loop.

The formulas for $\mathrm{VBI}(i, j)$ and $\mathrm{VM}(i, j)$ are the following:

$$\mathrm{VBI}(i, j) = \min_{\substack{i < i' < j' < j \\ i' - i + j - j' > 2}} \{\mathrm{ebi}(i, j, i', j') - V(i', j')\}$$

$$\mathrm{VM}(i, j) = \min_{i+1 < k < j-1} \{W(i+1, k) + W(k+1, j-1)\}$$

A Revised Loop Dependent Energy Minimization Algorithm Zuker's algorithm considers some additional features in the folding process.

First, for the hairpin loop whose length is greater than 3, the so-called *terminal mismatched pairs* are considered; that is, if base pair $i \cdot j$ closes a hairpin loop, then the bases $i+1$ and $j-1$ have some effects on the free energy of the hairpin loop. In addition, specific tetraloops, that is, hairpin loops with four bases, are assigned enhanced stability. These tetraloops are known to be more stable or to be important in stabilizing tertiary structures (Antao and Tinoco 1992; Lehnert et al. 1996; Mathews

et al. 1999). Furthermore, there are special hairpin loops—for instance, the hairpin with three single *G*s, with their own energy rules. These considerations have been incorporated into the values of $\mathrm{eh}(i, j)$.

It has also been shown by experiments that the single bases also can stabilize the RNA secondary structure. Suppose base i and base j form a base pair, and base $i+1$ or base $j+1$ is a single base, then base $i+1$ or base $j+1$ is called a *3′-dangling base*; if base $i-1$ or base $j-1$ is a single base, then base $i-1$ or base $j-1$ is called a *5′-dangling base*. Because single bases adjacent to helices are assumed to stack if the stacking is favorable, the algorithm considers that all dangling energies are nonpositive.

The assumption that the contribution of a multiple loop is zero is too far from reality. Biologists have shown that the multiple loops indeed play an important role in constructing RNA secondary structure. A general energy function for multiple loops may require exponential computing time. Zuker's algorithm uses a linear energy function to calculate the energy of multiple loops,

$$e(k\text{-loop}) = a + bk' + ck, \quad \text{where } k > 2$$

where a, b, and c are non-negative constants, k' is the number of single bases in the multiple loop.

To implement this equation, $W'(i, j)$ is used; the meaning of $W'(i, j)$ is almost the same as $W(i, j)$, except that it treats the exterior loops of optimal structures as multiple loops with appropriate penalties. W' also considers the special rules for some hairpin loops and dangling stabilizing energies.

$$W'(i, j) = \min \begin{cases} W'(i+1, j) + b, & (1) \\ \mathrm{ed}(i+1, j, i) + V(i+1, j) + b + c, & (2) \\ W'(i, j-1) + b, & (3) \\ \mathrm{ed}(i, j-1, j) + V(i, j-1) + b + c, & (4) \\ V(i, j) + c, & (5) \\ \mathrm{ed}(i+1, j-1, i) + \mathrm{ed}(i+1, j-1, j) + V(i+1, j-1) + 2b + c, & (6) \\ \min\limits_{i \le k < j} \{ W'(i,k) + W'(k+1, j) \} & (7) \end{cases}$$

where ed is the energy of the single dangling base: $\mathrm{ed}(i, j, k)$ means that base k is dangling on base pair $i \cdot j$. In this new equation, three more cases are added due to the consideration of dangling bases, cases (2), (4), and (6). In actuality, these cases

just refine the equation for $W(i, j)$. In addition, b and c are added as appropriate to match the linear penalty for multiple loop.

To understand this equation, consider the optimal structure of $r_i, \ldots, r_j$.

- In the optimal structure, if base i does not pair to any other base and does not stack on a base pair, then we have $W'(i, j) = W'(i+1, j) + b$, where b is the penalty contribution of base i.
- In the optimal structure, if base i does not pair to any other base but stacks on base pair $(i+1) \cdot j$, then we have $W'(i, j) = \text{ed}(i+1, j, i) + V(i+1, j) + b + c$, where b is the penalty contribution of base i and c is the penalty contribution of base pair $(i+1) \cdot j$.
- In the optimal structure, if base j does not pair to any other base and does not stack on a base pair, then we have $W'(i, j) = W'(i, j-1) + b$, where b is the penalty contribution of base j.
- In the optimal structure, if base j does not pair to any other base but stacks on base pair $i \cdot (j-1)$, then we have $W'(i, j) = \text{ed}(i, j-1, j) + V(i, j-1) + b + c$, where b is the penalty contribution of base j and c is the penalty contribution of base pair $i \cdot (j-1)$.
- In the optimal structure, if base i and base j form a base pair $i \cdot j$, then $W'(i, j) = V(i, j) + c$, where c is the penalty contribution of base pair $i \cdot j$.
- In the optimal structure, if base i and base j do not pair with any base and they both stack on base pair $(i+1) \cdot (j-1)$, then $W'(i, j) = \text{ed}(i+1, j-1, i) + \text{ed}(i+1, j-1, j) + V(i+1, j-1) + 2b + c$, where $2b$ are the penalty contribution of base i and base j and c is the penalty contribution of base pair $(i+1) \cdot (j-1)$.
- If none of the above is true, then the optimal structure will have at least two branches and we choose the structure with minimum energy.

Therefore we have

$$W'(i, j) = \min_{i \leq k < j} \{W'(i, k) + W'(k+1, j)\}$$

The equation for $V(i, j)$ is exactly the same as before.

$$V(i, j) = \min\{\text{eh}(i, j), \text{es}(i, j) + V(i+1, j-1), \text{VBI}(i, j), \text{VM}(i, j)\}$$

Recall that $\text{eh}(i, j)$ is for hairpin loops, $\text{es}(i, j)$ is for stacked pairs, $\text{VBI}(i, j)$ is for bulge or interior loops, and and $\text{VM}(i, j)$ is for multiple loops. The computation for

eh(i, j), es(i, j), and VBI(i, j) has no change. However, the computation for VM(i, j) is now more complicated due to the stacking energy of single bases.

$$\mathrm{VM}(i,j) = a + c + \min\{e_1(i,j) + e_2(i,j) + e_3(i,j) + e_4(i,j)\}$$

where

$$e_1(i,j) = \min_{i<k<j-1}\{W'(i+1,k) + W'(k+1,j-1)\}$$

$$e_2(i,j) = b + \mathrm{ed}(i,j,i+1) + \min_{i+1<k<j-1}\{W'(i+2,k) + W'(k+1,j-1)\}$$

$$e_3(i,j) = b + \mathrm{ed}(i,j,j-1) + \min_{i<k<j-2}\{W'(i+1,k) + W'(k+1,j-2)\}$$

$$e_4(i,j) = 2b + \mathrm{ed}(i,j,i+1) + \mathrm{ed}(i,j,j-1) + \min_{i+1<k<j-2}\{W'(i+2,k) + W'(k+1,j-2)\}$$

Because $i \cdot j$ closed a multiple loop, in the equation for VM(i, j) we have a penalty a for this multiple loop and a penalty c for base pair $i \cdot j$.

The functions $e_1(i,j)$, $e_2(i,j)$, $e_3(i,j)$, and $e_4(i,j)$ are obtained by considering various cases of dangling bases. When base $i+1$ and base $j-1$ are not stacked on $i \cdot j$, we have $e_1(i,j)$. When base $i+1$ but not $j-1$ is stacked on $i \cdot j$, we have $e_2(i,j)$. When base $j-1$ but not base $i+1$ is stacked on $i \cdot j$, we have $e_3(i,j)$. When both base $i+1$ and base $j-1$ are stacked on $i \cdot j$, we have $e_4(i,j)$.

The order to compute $W'(i,j)$ and $V(i,j)$ is for j to increase from 1 to n, and for i to decrease from j to 1. Once $W'(i,j)$ and $V(i,j)$ are computed, a linear array $W_5(i) = W(1,i)$, where $1 \le i \le n$, is computed. Finally, $W(1,n) = W_5(n)$ contains the energy value of the optimal structure. The computation of $W_5(i)$ is necessary because $W'(1,n)$ treats exterior loops as multiple loops with penalty, whereas our energy rule for exterior loops is to assign them zero energy. The equation for $W_5(i)$ is obtained by considering that dangling bases will contribute to the stability of the structure and that the energy for an exterior loop is zero.

$$W_5(0) = W_5(n+1) = 0$$

$$W_5(i) = \min\{W_5(i-1), W_5^1(i), W_5^2(i), W_5^3(i), W_5^4(i)\}$$

where

$$W_5^1(i) = \min_{0<k<i}\{W_5(k) + V(k+1,i)\}$$

$$W_5^2(i) = \min_{0 \le k < i} \{W_5(k) + \text{ed}(k+2, i, k+1) + V(k+2, i)\}$$

$$W_5^3(i) = \min_{0 \le k < i} \{W_5(k) + \text{ed}(k+1, i-1, i) + V(k+1, i-1)\}$$

$$W_5^4(i) = \min_{0 \le k < i} \{W_5(k) + \text{ed}(k+2, i-1, k+1) + \text{ed}(k+2, i-1, i) + V(k+2, i-1)\}$$

Time Complexity The time complexity of a direct implementation of this algorithm is $O(n^4)$, as we need $O(n^4)$ to compute VBI(i, j), the bulge or interior loop energy. Waterman and Smith (1986) and Lyngø et al. (1999) have used alternative equations to compute the bulge or interior loop energy by considering the lengths of the loops, which reduced the computation time to $O(n^3)$. Therefore, the time complexity of Zuker's algorithm is $O(n^3)$. With this time complexity, this algorithm can easily handle relatively long sequences.

Suboptimal Structures Another useful feature of Zuker's algorithm is that along with the optimal secondary structure, it can also produce suboptimal secondary structures (Zuker 1989). The biologically correct structure is often not the calculated optimal structure, but rather a structure within a few percent of the calculated minimum energy. Therefore suboptimal secondary structures will present alternative models for further investigation.

Suppose that E_{min} is the minimum folding energy. Let ΔE be a small energy increment; then, with some constraints, this algorithm can produce suboptimal secondary structures with energy value within ΔE of the optimal. Wuchty et al. (1999) proposed a different method to predict suboptimal foldings.

14.5 Phylogenetic Comparative Methods

Given a single RNA sequence, Zuker's folding algorithm (Zuker 1989) can produce optimal and suboptimal structures according to minimum energy criteria. However, in some cases, these structures do not closely resemble the real structure. Biologists use phylogenetic comparative methods on a set of related RNA sequences to determine common structures that are closer to the real structure.

One of the guiding principles in molecular biology is that structure is much more closely conserved than sequence. This principle also holds for RNA structures. The implication for secondary structure is that secondary structure is conserved even though sequence drift occurs. Thus when one base of a pair changes, we usually find that its partner also changes so as to conserve that base pair. This phenomenon is called a *compensatory base change* or *covariation*.

This leads to the following observation: if we have a set of related (meaning evolutionarily close) RNA sequences, then they are expected to have similar secondary structures. If a helical region appears in all the sequences, then it is more reliable than those that only appear in a small number of sequences. If a common secondary structure can be determined from the set, then it is probably more reliable than the one produced by using a thermodynamic algorithm on a single RNA sequence. This is the essence of the phylogenetic comparative method.

The framework of the phylogenetic comparative method is the following.

1. Select a set of related RNA sequences.
2. Perform multiple sequence alignment for the set of sequences.
3. Compute the covariation of any two columns of the alignment.
4. Determine the conserved helices and structures.

We need to select the sequences of homologous RNAs from different organisms among which the sequences vary. The choice of appropriate organisms for RNA structure comparisons requires a quantitative view of evolutionary relatedness. The molecules compared must be different enough to provide numerous instances of sequence variation with which to test pairing possibilities, yet they must not differ so much that residues cannot be aligned with confidence in the alignment step. In general, sequence similarities of 60–80 percent are favorable for inspecting the equivalent base pairings occurring in these sequences (James et al. 1989).

In order to accurately evaluate the covariations, a good multiple alignment of the sequences is required. Depending on the actual sequences, reliable alignments can be derived either from alignment software or from alignment software followed by manual adjustment. The result of this step is a multiple alignment of m RNA sequences:

$$
\begin{aligned}
R_1 &= r_1[1], r_1[2], r_1[3], \ \ldots, \ r_1[n] \\
R_2 &= r_2[1], r_2[2], r_2[3], \ \ldots, \ r_2[n] \\
R_3 &= r_3[1], r_3[2], r_3[3], \ \ldots, \ r_3[n] \\
\ldots & \qquad \ldots \qquad \ldots \\
R_m &= r_m[1], r_m[2], r_m[3], \ \ldots, \ r_m[n]
\end{aligned}
$$

For any i, $1 \leq i \leq n$, the set of bases $r_k[i]$ with $1 \leq k \leq m$, is referred to as a *column* of the alignment. Two columns of the alignment are said to be covarying when the variation in one column is echoed complementarily in the other column. The covariation of the two columns provides evidence for a conserved base pair.

The degree of covariation can be measured by several quantitative measures. One such measure is the *mutual information content* of two columns. For a single column i, let $f_i(r)$ be the frequency of the nucleotide r (where $r \in \{A, C, G, U\}$) in the ith column. Consider another column j, and let $f_{i,j}(r_1, r_2)$ be the joint frequency of the two nucleotides, r_1 from the ith column and r_2 from the jth column. If two columns vary independently, then $f_{i,j}(r_1, r_2)$ is roughly $f_i(r_1) \cdot f_j(r_j)$. Therefore we would expect $\log_2(f_{i,j}(r_1, r_2)/f_i(r_1)f_j(r_2))$ to be roughly zero. We are interested in measuring the divergence from independence. This leads to the definition of mutual information content (Chiu and Kolodziejczak 1991; Gutell et al. 1992), $M(i, j)$, between two different columns, i and j:

$$M(i,j) = \sum_{r_1, r_2 \in \{A, C, G, U\}} f_{i,j}(r_1, r_2) \log_2 \frac{f_{i,j}(r_1, r_2)}{f_i(r_1) f_j(r_2)}$$

$M(i, j)$ is maximized when both columns are highly variable and also completely correlated. Two advantages of this measure are that any types of correlations can be found and that correlations that are quantitatively low, but still significant, can also be found (Gutell et al. 1992).

Once the covariation measure is computed, possible conserved helices can be calculated (Winker et al. 1990; Chan et al. 1991) by searching the matrix $M(i, j)$. Finally, a conserved common secondary structure can be constructed from conserved helices (Chan et al. 1991; Gutell et al. 1992; Han and Kim 1993; Eddy and Durbin 1994; Le et al. 1995). One can apply Nussinov/Zuker algorithms for this step where the score being optimized is a function of covariation measure instead of the number of base pairs or of thermodynamic stacking energies. Nevertheless, this step normally requires manual intervention.

Phylogenetic comparative methods are currently considered the most effective methods, the advantage being that they produce reliable secondary structures. The disadvantage is that they depend on the availability of high-quality multiple alignments and that they normally involve manual intervention.

Phylogenetic comparative methods have also been used to determined RNA tertiary interactions (Gutell et al. 1992).

14.6 Stochastic Context-Free Grammar Method

A more recent approach uses a stochastic context-free grammar to model the common features of a set of RNA structures (Searls 1992, 1993; Eddy and Durbin 1994;

Sakakibara et al. 1994; Knudsen and Hein 1999). We will now give a brief description of this method.

One can view DNA and RNA sequences as sentences derived from a formal grammar (Searls, 1992, 1993). In the simplest kind of grammar, a regular grammar, sequences are derived from productions of the form $S \rightarrow aS$ and $S \rightarrow a$, where S is a non-terminal symbol and a is a terminal symbol. DNA sequences can be described by a regular grammar. Base pairing in RNA secondary structure can be described by a context-free grammar (CFG). Context-free grammars are more powerful than regular grammars because they allow additional forms of productions, such as these of the form $S \rightarrow SS$ and $S \rightarrow aSa$. The process of generating a sequence by repeatedly applying productions is called derivation. For CFGs, a derivation can be arranged in a tree structure called a parse tree.

Productions of the following forms can be used to model a RNA secondary structure: $S \rightarrow SS$, $S \rightarrow aSa$, $S \rightarrow aS$, and $S \rightarrow a$, where S is a non-terminal symbol and a is a terminal symbol. The production $S \rightarrow aSa$ describes base pairings; $S \rightarrow aS$ and $S \rightarrow a$ describe unpaired bases; and $S \rightarrow SS$ describes branching of secondary structure. A parse tree of these productions represents an RNA secondary structure. To model a family of RNA sequences, additional production forms will be needed.

A stochastic context-free grammar (SCFG) is a context-free grammar such that each production is associated with a probability value. The probability of a derivation (parse tree) can be calculated as the product of the probabilities of the productions producing the derivation. The probability of a sequence S under SCFG G is the sum of probabilities over all possible derivations of G that produce the sequence (Sakakibara et al. 1994):

$$\begin{aligned}\text{Prob}(s \mid G) &= \sum_{\text{all derivations } d} \text{Prob}(S_0 \overset{d}{\Rightarrow} s \mid G) \\ &= \sum_{\alpha_1,\ldots,\alpha_n} \text{Prob}(S_0 \Rightarrow \alpha_1 \mid G) \times \text{Prob}(\alpha_1 \Rightarrow \alpha_2 \mid G) \times \cdots \times \text{Prob}(\alpha_n \Rightarrow s \mid G)\end{aligned}$$

Because CFGs are ambiguous grammars, a sequence can be derived by more than one derivation or parse tree. For a single RNA, different parse trees usually represent different secondary structures. Because an SCFG assigns a probability to each parse tree, it has the advantage that the most likely parse tree for a given sequence can be determined. Intuitively, this most likely parse tree corresponds to the correct secondary structure if the SCFG G models the RNA sequences correctly.

In order to apply SCFG to RNA secondary structure prediction, we need to solve the following problems:

1. Given a set of example sequences/structures, determine the optimal SCFG. This includes the determination of the productions and the estimation of optimal probability parameters.

2. Given a SCFG G and an RNA sequence R, determine the most likely parse tree that corresponds to the correct RNA secondary structure.

The first problem can be solved by the inside-outside algorithm (Lari and Young 1990; Baker 1979), which is an expectation maximization (EM) algorithm that calculates maximum likelihood estimates of an SCFG's parameters based on training data. The second problem can be solved by a dynamic programming procedure similar to the Viterbi algorithm for hidden Markov models (HMMs) (Rabiner 1989).

Given a set of RNA sequences, Eddy and Durbin (1994) use multiple sequence alignment to construct a model (SCFG). They then iteratively reestimate the model by an EM algorithm. With a new model generated, a new alignment can be constructed. This process is repeated until the model stabilizes.

Sakakibara et al. (1994) proposed a method where an initial grammar instead of a multiple alignment is used. Their algorithm is based on a tree grammas and is more efficient than the inside-outside algorithm.

One can view SCFG method as an automated phylogenetic comparative method. Durbin et al. (1989) suggested that currently this aspect of the SCFG method is mostly theoretical rather than of practical interest.

14.7 Conclusions

RNA secondary structure prediction is an important problem in molecular biology. Thermodynamic energy minimization methods have been used to predict secondary structures from a single RNA sequence. Phylogenetic comparative methods have been used to determine secondary structures from a set of homologous RNAs whose sequences can be reliably aligned. One interesting problem is how to combine efficiently those two approaches, though there seems to be some inherent difficulties. Another challenging problem is to predict RNA tertiary structures. Although some initial attempts have been made, those problems have not been solved satisfactorily.

References

Antao, V., and Tinoco, I. J. (1992). Thermodynamic parameters for loop formation in RNA hairpin tetraloops. *Nucl. Acids Res.* 20(4): 819–824.

Baker, J. (1979). Trainable grammars for speech recognition. In *Speech Communication Papers for the 97th Meeting of the Acoustical Society of America*, 547–550.

Cech, T., and Bass, B. (1988). Biological catalysis by RNA. *Annual Review of Biochemistry* 55: 599–629.

Chan, L., Zuker, M., and Jacobson, A. (1991). A computer method for finding common base paired helices in aligned sequences: Application to the analysis of random sequences. *Nucl. Acids Res.* 19(2): 353–358.

Chen, J., Le, S., and Maizel, J. (2000). Prediction of common secondary structures of RNAs: A genetic algorithm approach. *Nucl. Acids Res.* 28(4): 991–999.

Chiu, D., and Kolodziejczak, T. (1991). Inferring consensus structure from nucleic acid sequences. *Computer Applications in the Biosciences* 7: 343–352.

Durbin, R., Eddy, S., Krogh, A., and Mitchison, G. (1998). RNA structure analysis. In *Biological Sequence Analysis*, pages 260–289. Cambridge, UK: Cambridge University Press.

Eddy, S., and Durbin, R. (1994). RNA sequence analysis using covariance models. *Nucl. Acids Res.* 22(11): 2079–2088.

Freier, S., Kierzek, R., Jaeger, J., Sugimoto, N., Caruthers, M., Neilson, T., and Turner, D. (1986). Improved free-energy parameters for predictions of RNA duplex stability. *Proc. Nat. Acad. Sci. USA* 83: 9373–9377.

Gutell, R., Larsen, N., and Woese, C. (1994). Lessons from an evolving rRNA: 16S and 23S rRNA structures from a comparative perspective. *Microbiology Review* 58: 10–26.

Gutell, R., Power, A., Hertz, G., Putz, E., and Stormo, G. (1992). Identifying constraints on the higher-order structure of RNA: Continued development and application of comparative sequence analysis methods. *Nucl. Acids Res.* 20(21): 5785–5795.

Han, K., and Kim, H. (1993). Prediction of common folding structures of homologous RNAs. *Nucl. Acids Res.* 21(5): 1251–1257.

James, B., Olsen, G., and Pace, N. (1989). Phylogenetic comparative analysis of RNA secondary structure. *Methods Enzymol.* 180: 417–425.

Knudsen, B., and Hein, J. (1999). RNA secondary structure prediction using stochastic context-free grammars and evolutionary history. *Bioinformatics* 15(6): 446–454.

Lari, K., and Young, S. (1990). The estimation of stochastic context-free grammars using the inside-outside algorithm. *Comput. Speech Lang.* 4: 35–56.

Le, S., Zhang, K., and Maizel, J. (1995). A method for predicting common structures of homologous RNAs. *Comput. Biomed. Res.* 28: 53–66.

Le, S., and Zuker, M. (1990). Common structure of the 5′ non-coding RNA in enteroviruses and rhinoviruses—thermodynamical stability and statistical significance. *J. Mol. Biol.* 216: 729–741.

Lehnert, V., Jaeger, L., Michel, F., and Westhof, E. (1996). New loop-loop tertiary interactions in self-splicing introns of subgroup ic and id: A complete 3D model of the tetrahymena thermophila ribozyme. *Chem. Biol.* 3: 993–1009.

Lyngø, R., Zuker, M., and Pedersen, C. (1999). Fast evaluation of internal loops in RNA secondary structure prediction. *Bioinformatics* 15(6): 440–445.

Mathews, D. H., Sabina, J., Zuker, M., and Turner, D. H. (1999). Expanded sequence dependence of thermodynamic parameters improves prediction of RNA secondary structure. *J. Mol. Biol.* 288: 911–940.

McCaskill, J. (1990). The equilibrium partition and base pair binding probabilities for RNA secondary structure. *Biopolymers* 29: 1105–1119.

Nussinov, R., Pieczenik, G., Griggs, J., and Kleitman, D. (1978). Algorithms for loop matchings. *SIAM J. Appl. Math.* 35(1): 68–82.

Pipas, J., and McMahon, J. (1975). Method for predicting RNA secondary structure. *Proc. Nat. Acad. Sci. USA* 72(6): 2017–2021.

Rabiner, L. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proc. IEEE* 77(2): 257–286.

Rivas, E., and Eddy, S. (1999). A dynamic programming algorithm for RNA structure prediction including pseudoknots. *J. Mol. Biol.* 285: 2053–2068.

Sakakibara, Y., Brown, M., Hughey, R., Mian, I., Sjölander, K., Underwood, R., and Haussler, D. (1994). Stochastic context-free grammar for tRNA modelling. *Nucl. Acids Res.* 22(23): 5112–5120.

Sankoff, D. (1985). Simultaneous solution of the RNA folding, alignment and protosequence problems. *SIAM J. Appl. Math* 45(5): 810–824.

Sankoff, D., Kruskal, J., Mainville, S., and Cedergren, R. (1983). Fast algorithms to determine RNA secondary structures containing multiple loops. In *Time Warps, String Edits, and Macro-Molecules: The Theory and Practice of Sequence Comparison*, Sankoff, D., and Kruskal, J., eds., chapter 3. Reading, MA: Addison-Wesley.

Searls, D. (1992). The linguistics of DNA. *American Scientist* 80: 579–591.

Searls, D. (1993). The computational linguistics of biological sequences. In *Artificial Intelligence and Molecular Biology*, 47–120. Menlo Park, Calif.: AAAI Press.

Shapiro, B., and Wu, J. (1996). An annealing mutation operator in the genetic algorithms for RNA folding. *CABIOS* 12(3): 171–180.

Tinoco, I., Uhlenbeck, O., and Levine, M. (1971). Estimation of secondary structure in ribonucleic acids. *Nature* 230: 362–367.

Turner, D., Sugimoto, N., and Freier, S. (1988). RNA structure prediction. *Ann. Rev. Biophys. Biophys. Chem.* 17: 167–192.

Turner, D., Sugimoto, N., Jaeger, J., Longfellow, C., Freier, S., and Kierzek, R. (1987). Improved parameters for prediction of RNA structure. *Cold Spring Harb. Symp. Quant. Biol.* 52: 123–133.

Wang, Z., and Zhang, K. (1999). Finding common RNA secondary structures from RNA sequences. In *Proceedings of the Tenth Symposium on Combinatorial Pattern Matching*, Crochemore, M., and Paterson, M., eds. Springer-Verlag's Lecture Notes in Computer Science 1645, 258–269. Berlin: Springer.

Waterman, M. (1978). Secondary structure of single-stranded nucleic acids. *Studies in Foundations & Combinatorics, Advances in Mathematics Supplementary Studies* 1: 167–212.

Waterman, M., and Smith, T. (1986). Rapid dynamic programming algorithms for RNA secondary structure. *Advances in Applied Mathematics* 7: 455–464.

Winker, S., Overbeek, R., Woese, C., Olsen, G., and Pfluger, N. (1990). Structure detection through automated covariance search. *CABIOS* 6: 365–371.

Wuchty, S., Fontana, W., Hofacker, I. L., and Schuster, P. (1999). Complete suboptimal folding of RNA and the stability of secondary structures. *Biopolymers* 49: 145–165.

Zuker, M. (1986). RNA folding prediction: The continued need for interaction between biologists and mathematicians. *Lectures Math. Life Sci.* 17: 86–123.

Zuker, M. (1989). On finding all suboptimal foldings of an RNA molecule. *Science* 244: 48–52.

Zuker, M. (2000). Calculating nucleic acid secondary structure. *Curr. Opin. Struct. Biol.* 10: 303–310.

Zuker, M., and Sankoff, D. (1984). RNA secondary structures and their prediction. *Bull. Math. Biol.* 46: 591–621.

Zuker, M., and Stiegler, P. (1981). Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information. *Nucl. Acids Res.* 9: 133–148.

15 Properties and Prediction of Protein Secondary Structure

Victor V. Solovyev and Ilya N. Shindyalov

15.1 Introduction

Secondary structure describes regular features of the main chain of protein molecules. Experimental investigations on polypeptides and small proteins suggest that secondary structure can form in isolation, implying the possibility of identifing rules for its computational prediction. Predicting secondary structure from the amino acid sequence alone is an important step toward our understanding of protein structure and function. It may provide a starting point for tertiary structure modeling, especially in the absence of a suitable homologues template structure, reducing the search space in simulation of protein folding. Advances in fold recognition (or threading) techniques (Fischer and Eisenberg 1996; Russel et al. 1996; Koretke et al. 1999; Di Francesco et al. 1999; Hargbo and Elofsson 1999; Fisher 2000) have been where secondary structure predictions and other protein characteristics were combined to suggest resemblance to a known fold. The predictions can also be used in various aspects of molecular biology research to provide clues about the functional properties of proteins under analysis. The goal in secondary structure prediction approaches is to extract the maximum information from the primary sequence in the absence of a tertiary structure model (King and Sternberg 1996). In this chapter we will deal with the description of secondary structure characteristics, assignment of secondary structure using known 3D coordinates, and prediction of secondary structure based on primary sequence.

15.2 Secondary Structure Elements in Globular Proteins

The term "secondary structure" was first introduced by Linderstrom-Lang (Linderstrom-Lang and Schnellman 1959) more than 40 years ago to describe regular structural patterns of polypeptide backbone independent of the types of amino acid side chains. There are three major types of secondary structure: α-helix, β-structure and β-turn.

The occurrence of different kinds of secondary structure can be seen in the Ramachandran plot (Ramachandran et al. 1966). In figure 15.1, one can see the distribution of φ and ψ angles for a total of 9,156 amino acid residues from 4,413 protein chains, based on crystallographic data. There are two areas where the density of points is high: (1) around $\varphi = -60°$ and $\psi = -60°$, which corresponds to the α-helix; (2) and around $\varphi = -90°$ and $\psi = 120°$, which corresponds to the β-structure.

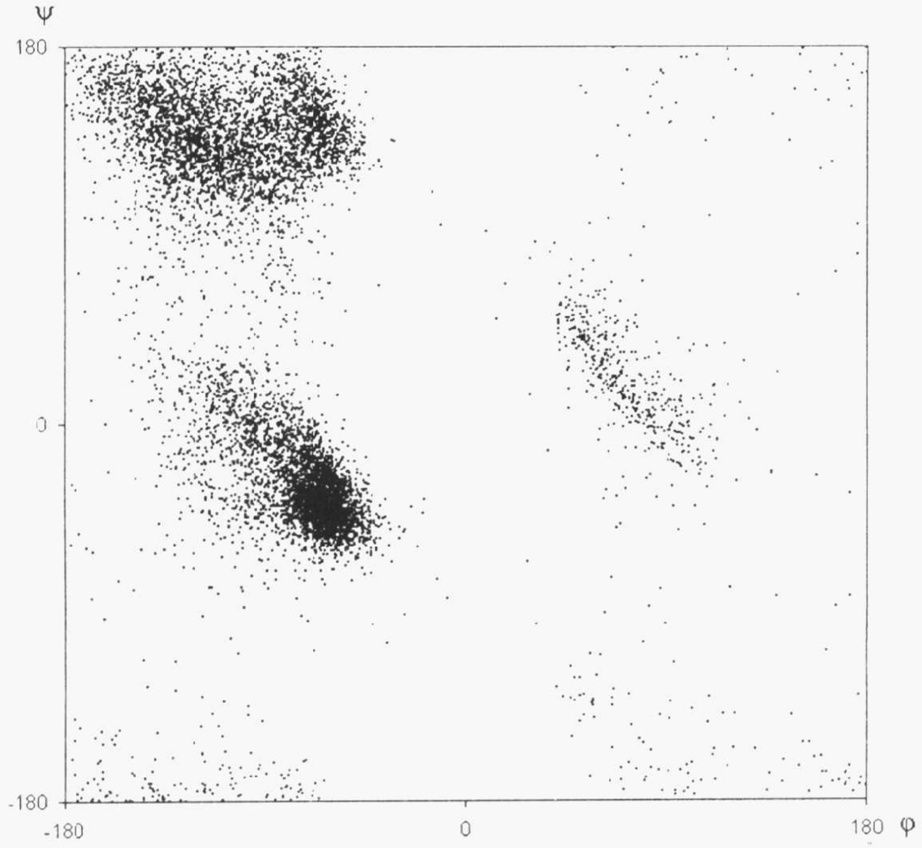


Figure 15.1
Distribution of main chain dihedral angles for random sample of 9,156 amino acids from 4,413 protein chains.

Figure 15.2 shows the Ramachandran plots for valine (a) and leucine (c) residues calculated for five thousand randomly chosen amino acids. The highly occupied areas of these plots have a good correspondence with low energy conformation of amino acid residues (Némethy and Scheraga 1977). All other residues have very similar plots, with the exception of glycine (figure 15.2D) and proline (figure 15.2B). Glycine has a much wider low-energy area because it does not have a C^{α} atom. Proline has its side chain covalently bound to backbone amine; hence its φ angle is limited to the range of $\varphi = -60^{\circ} \pm 20^{\circ}$.

φ and ψ angles associated with low-energy areas on energy plots are the angles observed in major types of secondary structure: α-helices and β-structures, as shown in figure 15.3. (Pauling et al. 1951). The largest area corresponding to β-structures can be seen as divided into four parts due to the limits of representation of continuous map. Two separate areas (α_R and α_L) correspond to right-handed and left-handed α-

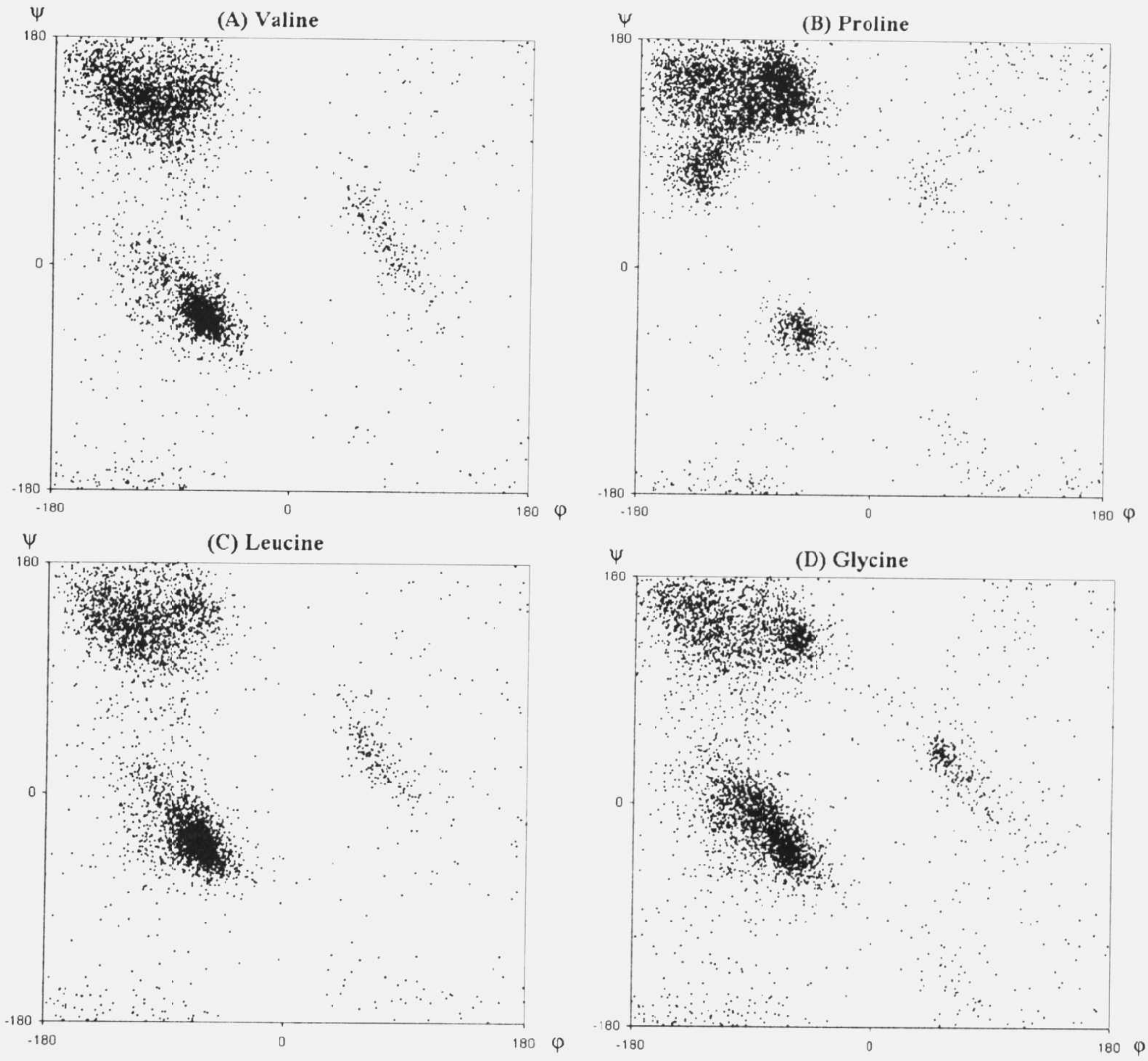


Figure 15.2
Comparison of main chain dihedral angles distributions for (A) Valine; (B) Proline; (C) Leucine; (D) Glycine.

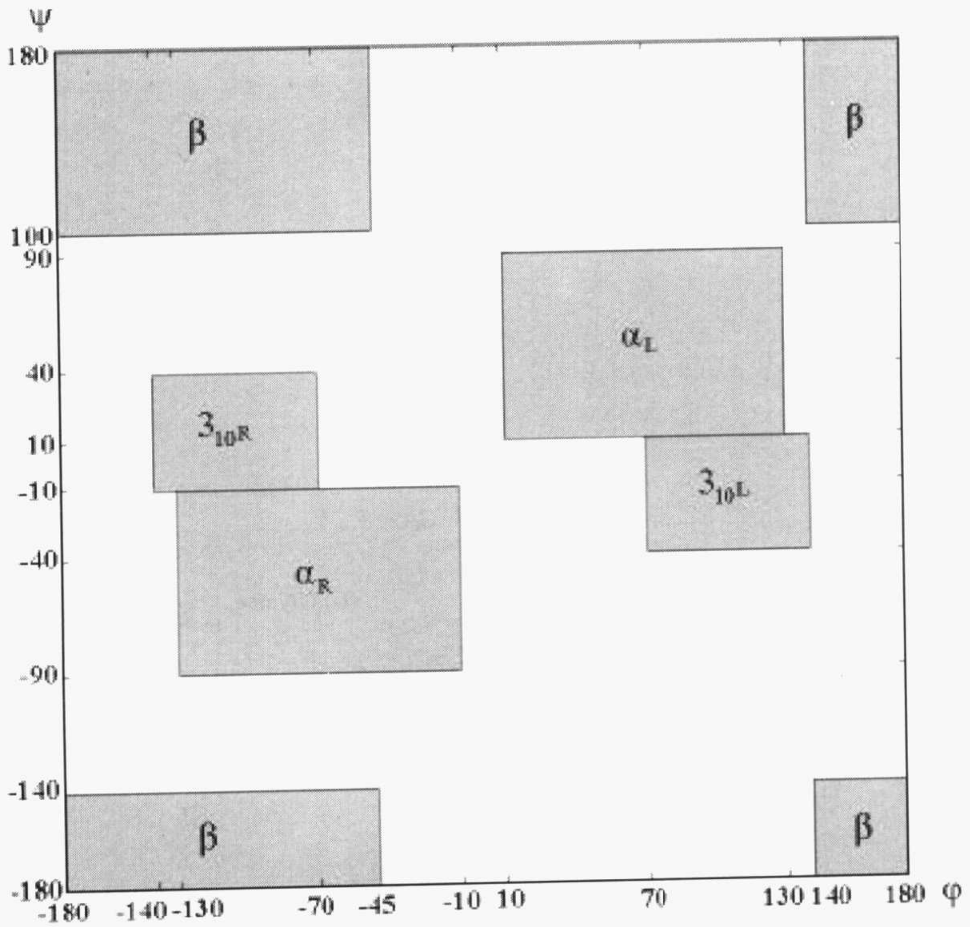


Figure 15.3
Conformational states in proteins (adapted from Némethy and Scheraga 1977). α_R and α_L, right and left α-helices; 3_{10^R} and 3_{10^L}, right and left 3_{10} helices (or β-turn of type III); β, β-structural conformation (extended polypeptide chain).

helices, respectively. Another two areas (3_{10^R} and 3_{10^L}) correspond to right-handed and left-handed 3_{10}-helices.

15.2.1 Helices

Regular backbone conformations with the elements maintaining nearly identical orientations (the same φ and ψ angles) can be described as helices, which can be described by the shift along the helical axis (d), the number of residues per turn (n), and the distance (r) from the specific residue location (e.g., C^α atom) to the helical axis.

Figure 15.4 shows the overall geometry and the schematic diagrams for 3_{10}- and α-helices. Protein helices are stabilized by hydrogen bonds between the amino and

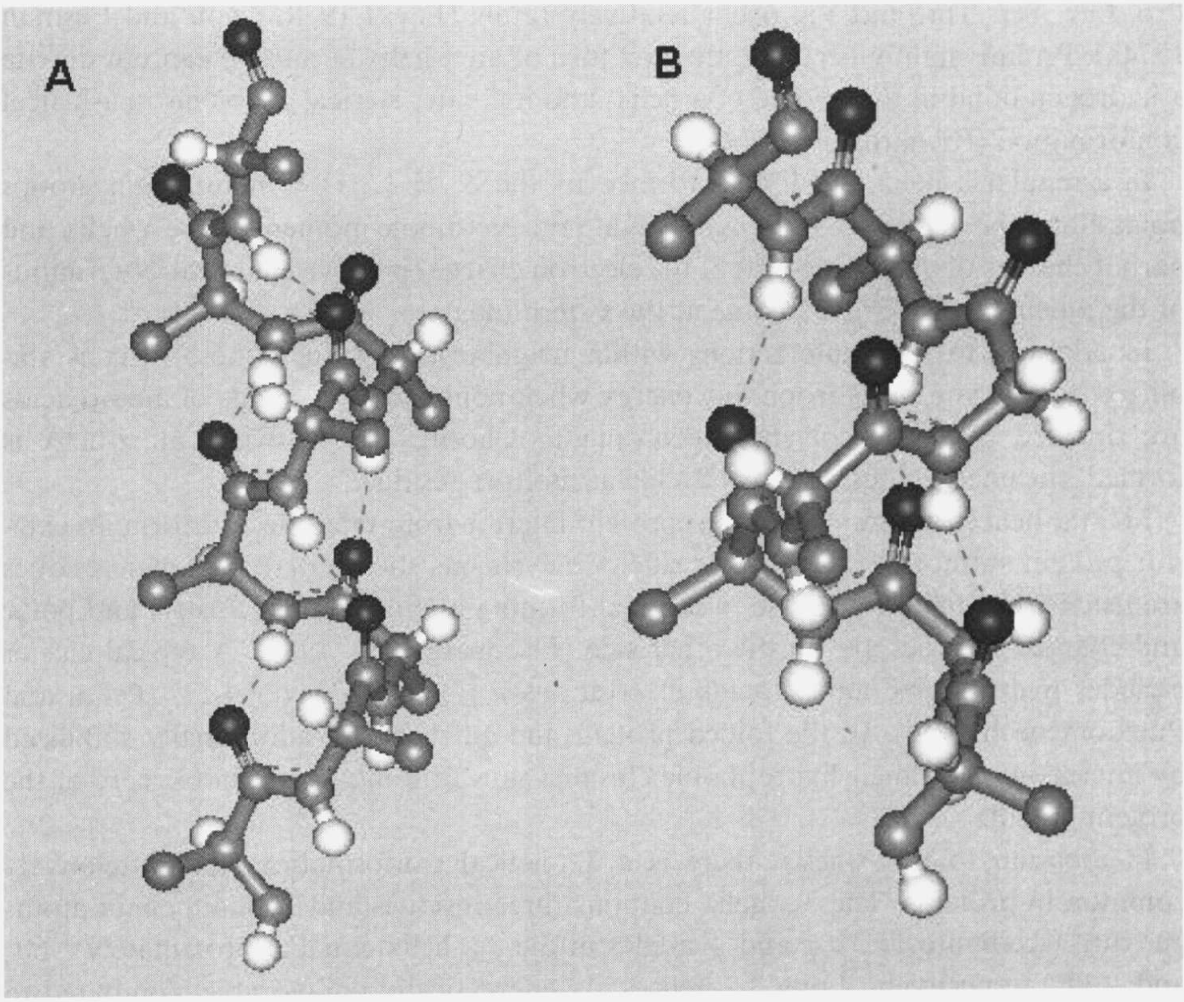


Figure 15.4
Helices with internal hydrogen bonds in proteins. (A) 3_{10}-helix; (B) α-helix.

carboxyl groups of the amino acid residue main chains: i, $i+3$ (3_{10}-helix); i, $i+4$ (α-helix); and i, $i+5$ (α-helix) (Schulz and Schirmer 1979).

The α-helix was first described by Pauling (Pauling et al. 1951), who modeled a stable polypeptide chain based on peptide unit geometry. There are about 35 percent of amino acids in the α-helical conformation, based on crystallographic data for 50 different proteins (Levitt 1978).

The average length of the α-helix is about 10–11 residues, which is approximately 17 Å, or three helical turns. The main chain angles in the α-helix are approximately $\varphi = \psi = -60°$ (Schulz and Schirmer 1979).

Some amino acid residues have small but distinguished preference to forming α-helical conformation. Ala, Glu, Leu, and Met are often found in α-helices, whereas

Pro, Gly, Ser, Thr, and Val occur relatively rarely (Levitt 1978; Chou and Fasman 1974a). Proline mainly occurs in the first turn of an α-helix because it can not donate a hydrogen bond in the middle of a helix, and it creates sterical problems in α-helical conformation (Richardson 1981).

In a regular α-helix, all dipoles formed by the N–H ... O–C main chain groups point along the helical axis. It creates total nonzero dipole moment of the α-helix and partial charges of about one-half of the electron charge (positive charge at N-terminus of the α-helix and negative charge at the C-terminus).

In addition to local interactions within neighboring residues, the α-helix is stabilized by the gain of hydrophobic energy when nonpolar side chains of amino acids are shielded from the solvent. According to Chothia (1976), when an α-helix is formed, the energy goes down by 2–3 kcal/mol per residue.

Most α-helices are immersed into protein interior from one side and form an exterior protein surface from the other side. Analysis has shown that nonpolar residues are usually located on one side of α-helix (forming a hydrophobic cluster) and polar and charged residues are on the other side (Eisenberg et al. 1984). A typical cluster includes hydrophobic amino acids in positions i, $i \pm 1$, $i \pm 3$, $i \pm 4, \ldots$ (Palau and Puigdoménech 1974). In the folded protein, the α-helices are additionally stabilized by interacting with their hydrophobic clusters, thus forming hydrophobic core of the protein globule.

In addition to the α-helix, there is a 3_{10}-helical conformation that is relatively common in proteins. The 3_{10}-helix contains three residues and 10 main chain atoms per turn (see figure 15.3). φ and ψ angles in the 3_{10}-helix equal approximately $-60°$ and $-30°$, respectively. Long 3_{10}-helices are energetically unfavorable, compared to α-helices; hence the majority of such helices contain no more than two turns. The 3_{10}-helix is often located at the C-terminus of an α-helix (Richardson 1981). Dipoles formed with hydrogen bonds in 3_{10}-helices are not parallel to the helical axis because the energy minimum is not reached. Furthermore, the side chains are placed into a sterically unfavorable conformation, where the residues i and $i + 3$ are not on the same line along the helical axis (see section 15.2.3) (Schulz and Schirmer 1979).

Several left α-helices have been found in proteins. The longest one occurs in alcohol-dehydrogenase and has three hydrogen bonds. A left α-helix containing six amino acids has been found in thermolysine (Kabsch and Sander 1983).

15.2.2 β-structures

The second common type of ordered polypeptide conformation (after the α-helix) is the β-structure. About 36 percent of amino acid residues in globular proteins are in β-structural state (Levitt 1978).

The φ and ψ main chain angles of the β-structure are spread widely in the upper-left corner of the Ramachandran plot (see figure 15.3). $\varphi = \psi = 180°$ corresponds to the allowed conformation on (φ, ψ)-map and represents the fully extended conformation of the polypeptide chain (Richardson 1981). When looking along the polypeptide framework, one can see that the neighboring side chain groups are pointing to the opposite directions. However, such fully extended conformation is favorable for polyglycine (figure 15.5) only. In the presence of other amino acids, the φ and ψ angles are slightly different.

The flat layer of β-strands (β-sheet) was first described by Pauling et al. (1951) as one of the structures with the maximum hydrogen bonding between the C=O and N–H groups of the main chain. There are two possible mutual arrangements of β-strands in the β-sheet with respect to polypeptide chain direction: parallel and antiparallel (figure 15.6).

Most β-sheets in proteins are not flat but have a left twist, if one looks along a single β-strand (Schulz and Schirmer 1979). Every β-strand can be considered a very extended left helix, which turns 60° per two residues (figure 15.7). The twist in β-structure allows for conformational stabilization, providing energetically favorable contacts between the side chains of neighboring β-strands and the optimal orientation of the hydrogen bonds (Richardson 1981). In addition, such β-sheets are capable of dense packing between them, as well as with α-helices

Parallel β-structures usually occur inside a protein. They are often surrounded by α-helices protecting them from the solvent on both sides. Parallel β-structures usually do not form layers with fewer than five β-strands. The φ and ψ angles in parallel β-strands are more regular than in antiparallel β-strands. This indicates that parallel β-sheets are less stable than the antiparallel. In β-strands, one side is typically exposed to solvent and the other side is embedded inside the protein. This results in the characteristic interchange of hydrophobic and polar amino acids in antiparallel β-sheets (figure 15.7). The area in a β-strand involved in H-bonding with another β-strand ranges from one to nine residues in antiparallel β-sheets and from one to five residues in parallel β-sheets, respectively. The number of β-strands in a β-sheet varies from two to thirteen. β-sheets of mixed parallel and antiparallel β-strands comprise less than 20 percent of all cases. Main chain angles are distorted in the mixed β-sheets because some β-strands are involved in both parallel and antiparallel interactions on the opposite sides (Richardson 1981).

Amino acids Val, Ile, Tyr, and Thr have a preference for the β-structural conformation, whereas Glu, Gln, Lys, Asp, Pro, and Cys are rarely found in it (Levitt 1978). In general, β-structures are favored by bulky amino acids, which usually have limited

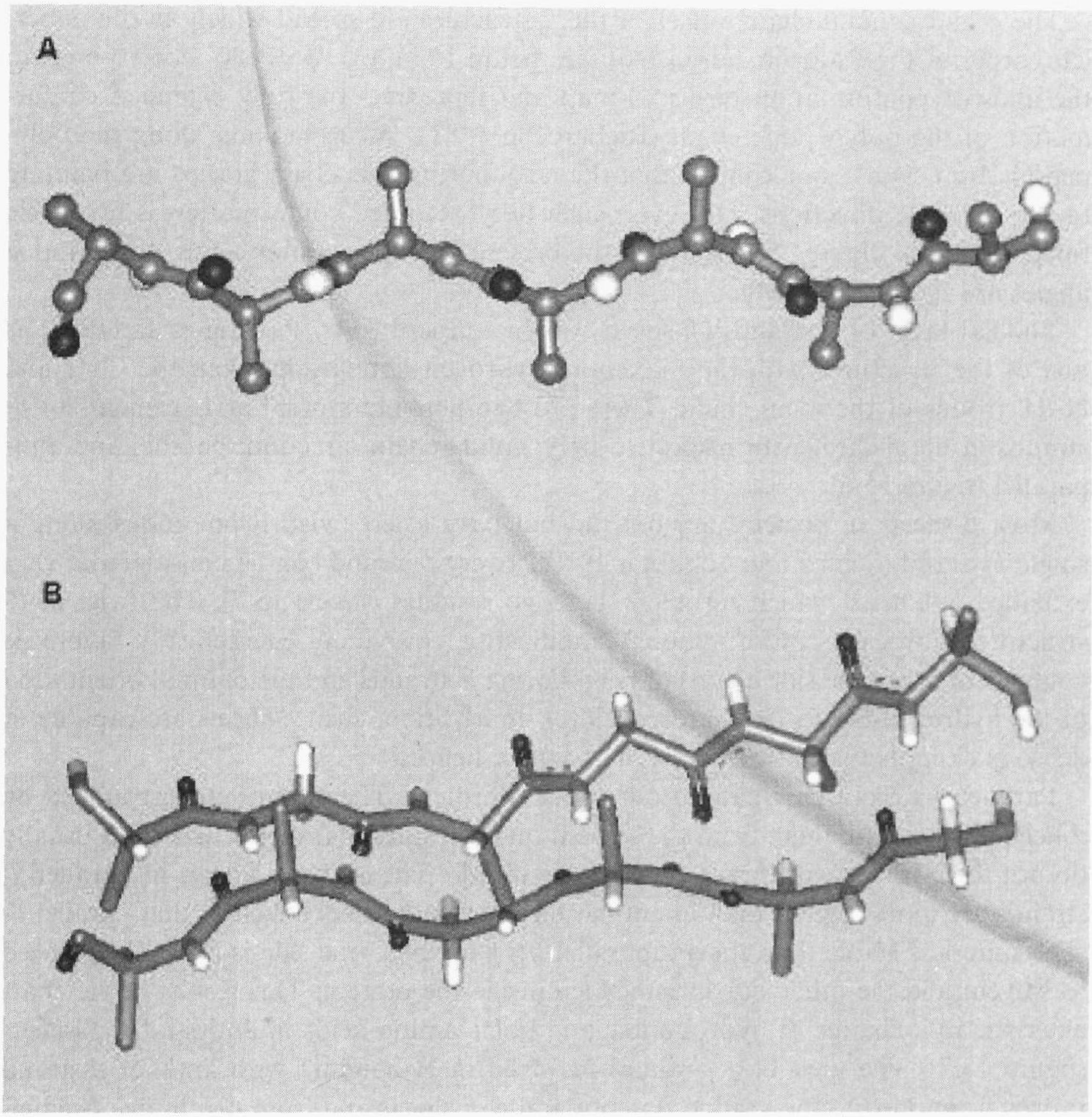


Figure 15.5
β-structural conformation. (A) β-strand geometry; (B) Interacting β-strands.

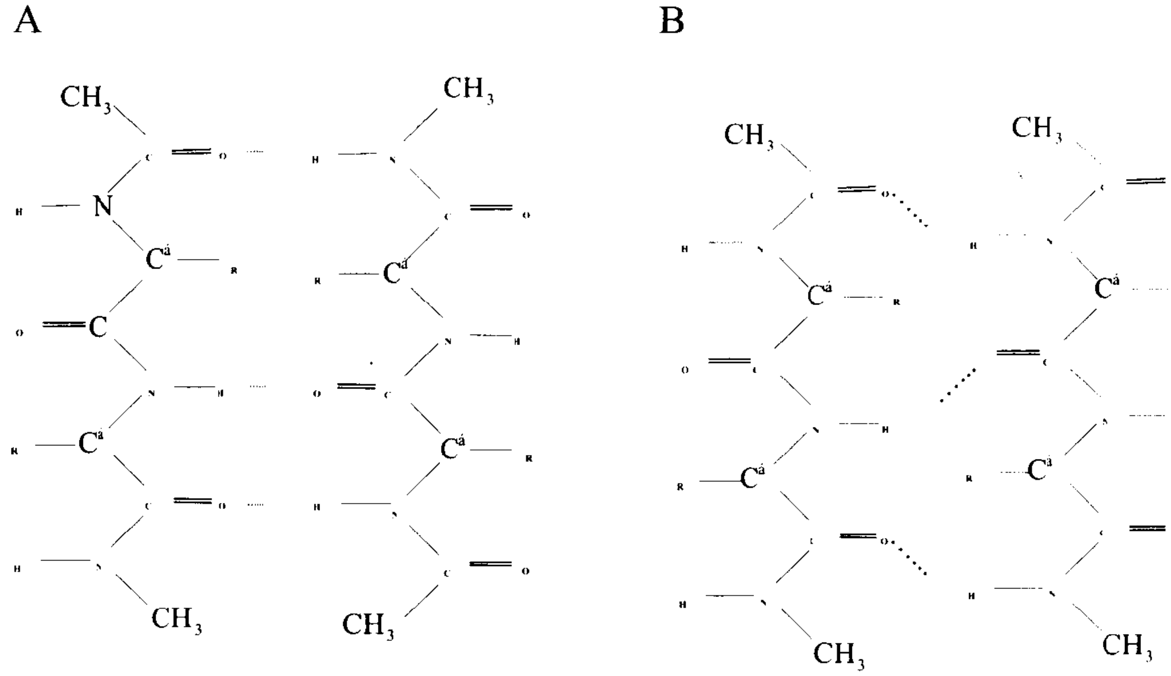


Figure 15.6
Features of β-structural conformation. (A) Antiparallel β-sheet; (B) Parallel β-sheet.

conformational flexibility due to branching at the C^{β} atom or a large aromatic group.

15.2.3 β-turns

After the exclusion of α-helices and β-structures, the rest of the polypeptide chain usually denotes as random coil or non-regular structure. Nevertheless, there are some fragments with recurring conformation within coiled structures. The most frequent one is the β-turn, which accounts for nearly 32 percent of all amino acid residues (Chou and Fasman 1977).

β-turn is a polypeptide fragment comprised of four consecutive amino acid residues in a region where the polypeptide chain changes direction roughly 180° (Chou and Fasman 1977). Conformation of the β-turn was first described in a theoretical study of three peptide units (or four neighboring residues) that are stabilized by a hydrogen bond between C=O of residue i and N–H of residue $i + 3$. Two most stable conformations were revealed, corresponding to type I and type II β-turns (figure 15.8), which represent 35 percent and 15 percent, respectively, of all β-turns. In addition, 15 percent of all turns are single turns of the 3_{10}-helix (type III β-turns), and 10 percent of all turns are mirror conformations of the type I–III turns.

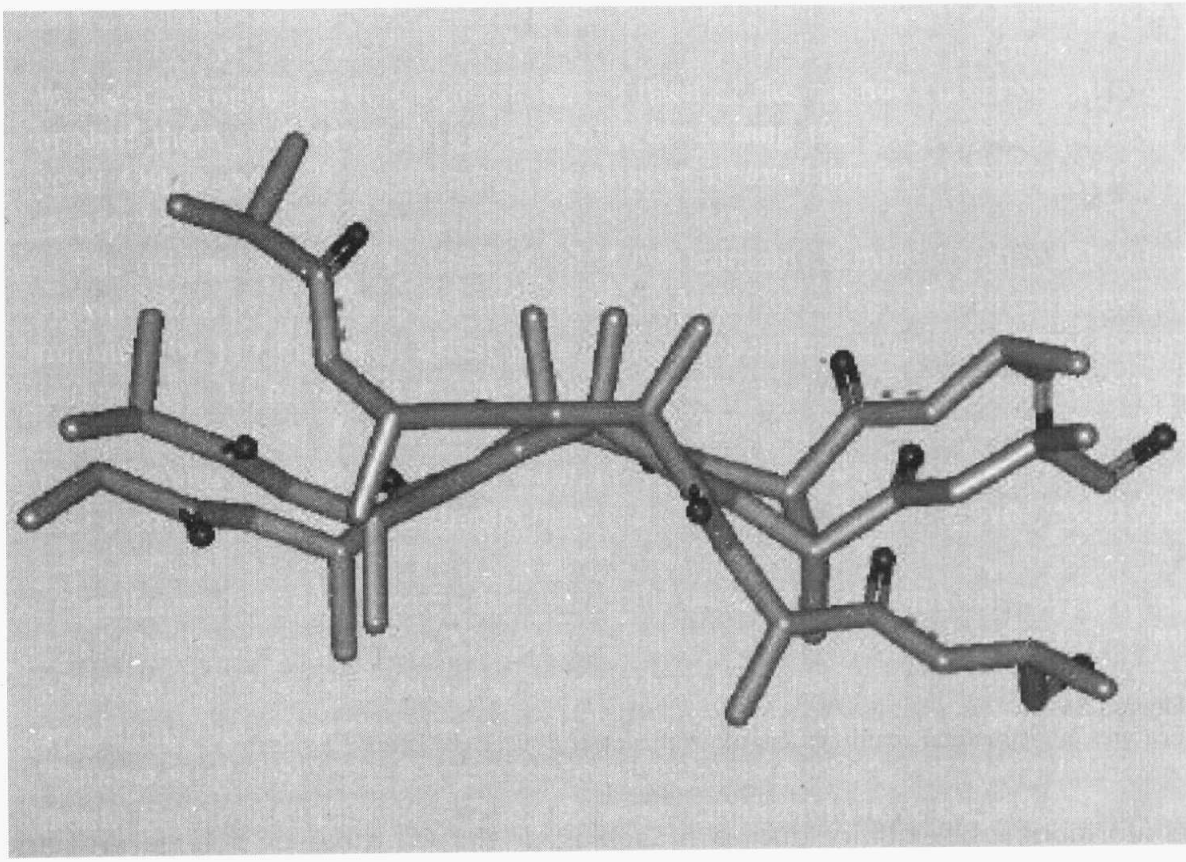


Figure 15.7
Twist in β-sheets.

Non-standard turns (or bends) containing no hydrogen bonds occur in 25 percent of cases. They can be recognized if the distance between the C^{β} atoms of residues i and $i + 3$ is less than 7 Å and those residues are not the part of an α-helix.

β-turns are usually located on the protein surface and contain many polar and charged amino acid side chains. Most turns contain glycine in the second or third position, where the absence of a side chain in glycine is favorable for the interaction among main chain atoms (Richardson 1981). Proline often occurs at the second position of turns. About two-thirds of Pro-Gly and Pro-Asp sequences in proteins with known 3D structures are located in the two middle residues of β-turns (Richardson 1981). Many β-turns connect neighboring fragments of secondary structures (α-α, α-β, and β-β). In these cases, the β-turn conformation can be stabilized by non-local interactions between the long secondary structures (Schulz and Schirmer 1979; Richardson 1981).

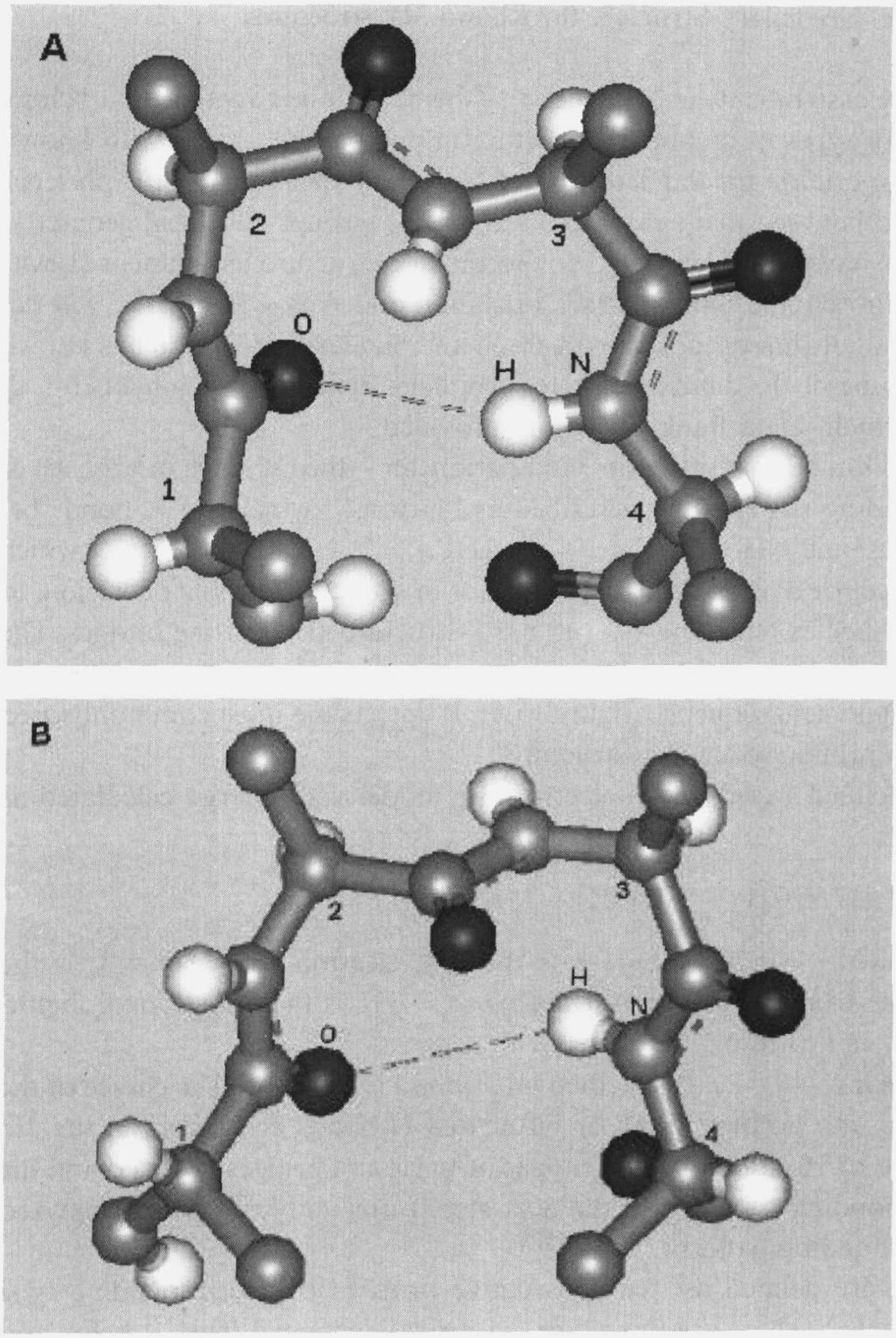


Figure 15.8
Type I and type II of β-turns. (A) Type I β-turn in tenascin (PDB ID 1TEN) at positions 815–818, length of H-bond 2.77 Å. (B) Type II β-turn in mannose permease (PDB ID 1PDO) at positions 84–87, length of H-bond 2.99 Å.

15.3 Assignment of Secondary Structure for Known 3D Structures

Secondary structure assignment can be defined as the detection of regions that belong to the various major types of secondary structure in polypeptide chains with known 3D structure. Major criteria for the detection of secondary structure are the presence of a hydrogen bond between main chain CO– and NH– groups and local geometry. Several approaches have been developed for secondary structure assignment (Levitt and Greer 1977; Kabsch and Sander 1983; Frishman and Argos 1995). We will describe in detail the algorithm proposed by Kabsch and Sander (1983), which is known as DSSP, by the name of the database where secondary structure assignments for all structures in the Protein Data Bank (PDB) are provided.

The DSSP algorithm takes a single decision parameter—the presence or absence of an H bond. H-bonding patterns are described as "n-turns," which are H-bonds between CO of residue i and NH of residue $i + n$, where $n = 3, 4, 5$; and "bridges," which are H-bonds, between residues not near each other in sequence. In this definition, α-helices can be described as repeating 4-turns and β-structure as repeating bridges. The results can be easily summarized and presented as a single character string corresponding to the amino acid sequence (figure 15.9). It detects the most commonly used representation of a protein secondary structure.

An H-bond is defined based on an electrostatic model with energy calculated as follows:

$$E = q_1 q_2 (1/r(\mathrm{ON}) + 1/r(\mathrm{CH}) - 1/r(\mathrm{OH}) - 1/r(CN)) \cdot f$$

with $q_1 = 0.42e$ and $q_2 = 0.20e$, where e is the unit electron charge, $r(\mathrm{AB})$ is the interatomic distance from A to B in Angstroms, $f = 332$ is the dimensional factor, and E is the energy in kcal/mol.

If the energy is below −0.5 kcal/mol, then an H-bond is assigned. The choice of the energy cutoff takes into account weak or bifurcated H-bonds and errors in the 3D coordinates. Figure 15.10 summarizes the types of turns and bridges detected and the corresponding H-bonding patterns. In the next step, turns and bridges are analyzed for cooperative H-bonding patterns.

Minimal helices are defined as two consecutive turns of the corresponding type (e.g., 3-turn for 3-helix). There are three types of helices defined: 3-helix, 4-helix, and 5-helix, corresponding to 3_{10}, α, and π helices, respectively. Minimal 4-helix of length 4 (starting at residue i) requires two 4-turns at residues $i - 1$ and i. Longer helices are built from the overlapping minimal helices. In the summary (figure 15.9), the helices are represented with letters G, H, and I for 3-, 4-, and 5-helices, respectively, bracketed with H-bonds (shown as ">" and "<").

(131L) Lysozyme_ (E.C.3.2.1.17)_Mutant

**** SECONDARY STRUCTURE DEFINITION BY THE PROGRAM DSSP, VERSION JUNE 1983 **** .

#	RESIDUE	AA	STRUCTURE	BP1	BP2	ACC	TCO	KAPPA	ALPHA	PHI	PSI	X-CA	Y-CA
1	1	M		0	0	59	0.000	0.0	360.0	360.0	150.6	43.4	-1.8
2	2	N	> -	0	0	68	-0.919	0.0	-82.9	-151.1	179.2	40.2	-0.[illegible]
3	3	I	H > S+	0	0	26	0.815	124.2	51.[illegible]	-56.6	-36.6	38.1	2.[illegible]
4	4	F	H > S+	0	0	75	0.938	113.4	42.[illegible]	-65.9	-47.8	40.2	3.[illegible]
5	5	E	H > S+	0	0	94	0.840	114.0	54.1	-65.5	-36.1	43.4	3.[illegible]
6	6	M	H X S+	0	0	0	0.951	111.9	41.5	-63.1	-53.2	41.8	4.7
7	7	L	H X S+	0	0	0	0.863	109.1	60.2	-64.6	-31.9	40.[illegible]	7.8
8	8	R	H X S+	0	0	91	0.883	107.7	44.5	-61.4	-33.9	43.7	8.[illegible]
9	9	I	H < S+	0	0	87	0.915	116.6	46.4	-73.4	-49.7	45.6	8.5
10	10	D	H < S+	0	0	19	0.823	125.0	28.7	-71.1	-34.1	43.0	10.4
11	11	E	H < S-	0	0	50	0.722	89.4	-156.0	-98.3	-29.3	42.7	13.[illegible]
12	12	G	< -	0	0	22	-0.242	25.7	-86.5	75.0	-167.5	46.1	13.0
13	13	L	+	0	0	41	-0.960	40.6	174.3	-141.3	125.6	46.6	13.7
14	14	R	E -A	28	0A	146	-0.998	17.7	-160.5	-131.3	133.6	47.1	17.1
15	15	L	E S+	0	0	60	0.454	74.1	58.9	-93.5	-3.9	47.3	17.[illegible]
16	16	K	E S-C	57	0B	124	-0.895	102.9	-81.4	-121.6	155.4	46.6	21.4
17	17	I	E +	0	0	29	-0.265	58.7	176.7	-49.7	135.1	43.7	23.2
18	18	Y	E -A	26	0A	23	-0.918	34.1	-108.5	-139.6	162.0	44.1	23.[illegible]
19	19	K	E -A	25	0A	123	-0.802	38.5	-139.8	-91.1	132.0	42.2	25.[illegible]
20	20	D	> -	0	0	47	-0.094	37.7	-78.1	-78.5	-168.2	40.8	22.[illegible]
21	21	T	T 3 S+	0	0	115	0.770	136.6	45.8	-59.5	-27.7	40.[illegible]	22.5
22	22	E	T 3 S-	0	0	77	0.373	124.4	-105.6	-94.2	-[illegible].5	37.6	24.[illegible]
23	23	G	S < S+	0	0	36	0.643	74.7	138.6	80.5	[illegible]2.9	39.3	27.[illegible]
24	24	Y	-	0	0	75	-0.815	60.7	-37.4	-101.3	[illegible]	37.[illegible]	[illegible]
25	25	Y	E +AB	19	34A	32	-0.325	56.1	153.1	-69.6	126.1	38.6	[illegible]
26	26	S	E -AB	18	32A	2	-0.839	23.3	-158.7	-144.2	175.2	39.[illegible]	[illegible]
27	27	I	E > - B	0	31A	0	-0.990	51.6	-1.4	-155.4	164.6	41.4	19.1
28	28	G	E 4 S-A	14	0A	0	-0.344	120.8	-2.3	61.3	-126.3	42.8	15.[illegible]

Figure 15.9
Sample of secondary structure assignment by DSSP (Kabsch and Sander 1983).

To describe a β-structure, two types of patterns are introduced: "ladder" and "sheet." A ladder is a set of one or more consecutive bridges of the same type. A sheet is one or more ladders connected by shared residues. Ladders and sheets are labeled alphabetically, parallel ladders by lower case, antiparallel by upper case, and sheets by upper case. Ladders of size 1 are labeled by "B," and longer ladders are labeled by "E" in the summary.

Additional rules were introduced to handle the secondary structure irregularities. For helices, it follows implicitly from the definition that two overlapping minimal helices offset by two or three residues are joined into one helix. For a β-structure, it was explicitly defined that allowance for β-bulges is such that two perfect ladders or bridges can be connected through a gap of one residue in one strand and four residues in the other strand. In the summary, all bulge-linked ladders are labeled with "E."

In addition to secondary structure, two geometrical features of polypeptide chains are also incorporated into the summary: "bend" and "chirality" (see figure 15.10 for details). The bend is assigned at sites where the polypeptide chain has an angle of

```
3-turn

       Notation            >     3     3     <
       Residues         -N-C-C--N-C-C--N-C-C--N-C-C-
                         H   O H   O H   O H   O
       H-Bond                \___________/

4-turn

       Notation            >     4     4     4     <
       Residues         -N-C-C--N-C-C--N-C-C--N-C-C--N-C-C-
                         H   O H   O H   O H   O H   O
       H-Bond                \_________________/

5-turn

       Notation            >     5     5     5     5     <
       Residues         -N-C-C--N-C-C--N-C-C--N-C-C--N-C-C-N-C-C-
                         H   O H   O H   O H   O H   O     O
       H-Bond                \_________________/

Parallel bridge

       Notation                    x
       Residues         -N-C-C--N-C-C--N-C-C-
                         H   O H   O H   O
       H-bonds               \/    \/
                             /\    /\
                         H   O H   O H   O
       Residues         -N-C-C--N-C-C--N-C-C-
       Notation                    x

Antiparallel bridge

       Notation                    X
       Residues         -N-C-C--N-C-C--N-C-C-
                         H   O H   O H   O
       H-bonds               | |   | |
                             | |   | |
                         O   H O   H O   H
       Residues         -C-C-N--C-C-N--C-C-N-
       Notation                    X
```

Figure 15.10
Hydrogen bonding patterns (adapted from Kabsch and Sander 1983).

more than 70° between line segments connecting C^{α} atoms from residue i to $i-2$ and i to $i+2$. The chirality for residue i is defined as a sign of the dihedral angle built on C^{α} atoms from residues $i-1$, i, $i+1$, $i+2$.

A similar approach to secondary structure assignment was introduced by Frishman and Argos (1995) in the STRIDE algorithm, which extends the DSSP definition by adding information on backbone torsion angles. H-bonding criteria were also improved by incorporation of experimental information on hydrogen bonds geometry and refining energy function. This resulted in better approximation to the assignment of secondary structure provided by experimentalists who determined the 3D structure. For 226 proteins in a test data set, STRIDE assigns 58 percent of the proteins closer to experimental assignment compared to DSSP, whereas DSSP assigns only 31 percent closer than STRIDE. For 11 percent of the proteins, both computational assignments are in the same proximity to the experimental ones.

15.4 Computational Methods of Secondary Structure Prediction

15.4.1 Summary of Earlier Approaches

Methods for the prediction of secondary structure from amino acid sequence have been developed for more than 30 years. They can be grouped into the following basic groups: (1) those that are based on stereo-chemical rules (Lim 1974), (2) those that use a combination of rules and statistical parameters (Chou and Fasman 1974a, b, 1978), (3) those that apply physical models of secondary structure formation (Ptitsyn and Finkelstain 1983), (4) those that are based on statistical information and information theory (Nagano 1973; Garnier et al. 1978; Gibrat et al. 1987), (5) those that analyze sequence patterns (Cohen et al. 1983, 1986), (6) those that are based on discriminant analysis (Solovyev and Salamov 1991, 1994), (7) neural-network approaches (Qian and Sejnowski 1988; Holley and Karplus 1989; Zhang et al. 1992), (8) analyzing evolutionary conservation in multiple alignments (Zvelebil et al. 1987), (9) nearest-neighbor algorithms (Salzberg and Cost 1992), and (10) consensus prediction algorithms (Viswanadhan et al. 1991). There are also methods that combine characteristics and principles from several of these categories.

At the beginning of 1990, these methods had achieved about 65 percent accuracy in predicting secondary structure in three states: α-helices, β-strands, and coil segments. During the last decade, there has been much improvement in methods based on discriminant analysis, neural network, and nearest-neighbor based approaches; algorithms such as PHD (Rost and Sander 1993, 1994), NNSP (Salamov and Solovyev 1995), SSPAL (Salamov and Solovyev 1997), DSC (King and Sternberg 1996, 1997),

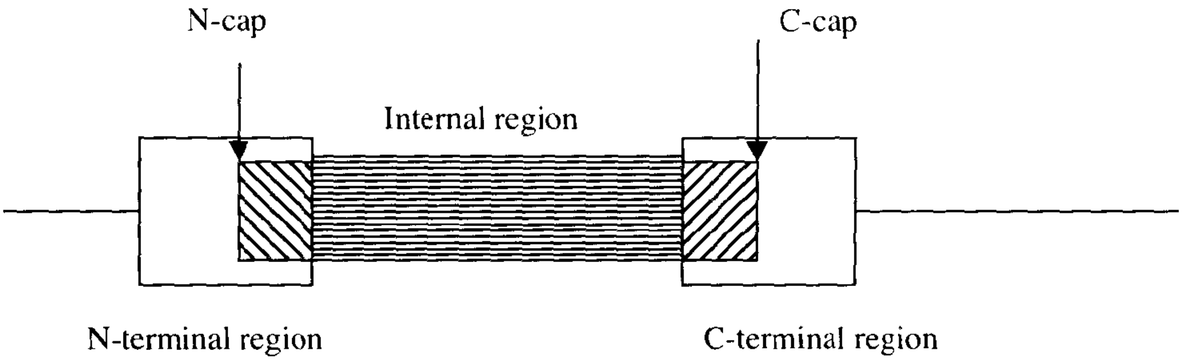


Figure 15.11
A model of α-helical (β-strand) segments.

alignment based nearest-neighbor (Ito et al. 1997; Frishman and Argos 1997), and PSI-PRED and its modifications (Jones 1999; Petersen et al. 2000) have yielded an accuracy generally ranging from 70 to 80 percent. In the following sections, we will describe the major features of these methods and discuss their performance.

15.4.2 Discriminant-analysis Approaches

The definition of secondary structure is somewhat imprecise and different authors may arrive at different secondary structure assignments for the same protein by using different assignment algorithms. For the purpose of modeling tertiary structures, it is more important to predict accurately the location of entire α-helix and β-strand segments than to reach higher single residue accuracy (Rost et al. 1994). Taking this into account, Solovyev and Salamov (1991, 1994) developed the Secondary Structure Prediction (SSP) program for the identification of whole α-helices and β-strands. They postulated that a particular conformation of protein segment can be determined by the combined action of three elements (figure 15.11): N-terminal, internal, and C-terminal regions, because the N- and C-terminal of α-helices and β-strands are formed by amino acids with specific properties (Richardson and Richardson 1988; Presta and Rose 1988) that are different from the characteristics of internal regions of these structures. The preference of these segments for the α-helix or β-strand conformation was estimated on the basis of the corresponding values of linear discriminant functions (LDF), which combined three characteristics of segments: average single-residue, a pair of residues preference parameters, and hydrophobic moment. The LDF were determined using linear discriminant analysis (Afifi and Azen 1979), described in detail in chapter 9 on a database of proteins of pairs of alternative classes: α-helical/non-α-helical and β-strand/non-β-strand segments. Because the mechanisms of formation of long and short secondary structures may differ during protein folding, and also to increase the

discriminating power of LDF, Solovyev and Salamov separately examined the cases of long and short structures. Thus, four pairs of alternative classes were considered for short and long α-helical and β-strand segments.

Characteristics Description Only three simple characteristics were used to assign a given amino acid region to one of the secondary structure classes.

The *singleton characteristic* is an average of single-residue preferences. Using a database of known protein structures, we calculated preferences (Chou and Fasman 1978) of all 20 types of amino acids to be in different part of α-helix and β-strand segments (N- and C-terminal, internal part, adjacent N- and C-terminal regions). For any segment, the average preference for α-helix or β-strand was computed as the sum of the individual preferences of all residues in the segment (15.1):

$$S = \frac{1}{L}\left(\sum_{i=l_1-m}^{l_1-1} S_i^{N_l} + \sum_{i=l_1}^{l_1+m-1} S_i^{N} + \sum_{i=l_1+m}^{l_2-m} S_i^{in} + \sum_{i=l_2-m+1}^{l_2} S_i^{C} + \sum_{i=l_2+1}^{l_2+m} S_i^{C_r}\right) \tag{15.1}$$

Here $L = l_2 - l_1 + 2m + 1$, l_1 and l_2 are the first and the last position of the segment; $m = 4$ or 3 for long α-helices and β-structures, respectively; $m = \mathrm{Int}[(l_2 - l_1 + 1)/2]$ for short segments; $S_i^{N_l}$, S_i^N, S_i^{in}, S_i^C, $S_i^{C_r}$ are the preference parameters of amino acid in position i that is adjacent to N-terminal, N-terminal, internal, C-terminal, or adjacent to the C-terminal part of the segment, respectively (Chou and Fasman 1978):

$$S^k(j) = \frac{P^k(j)}{P(j)} \tag{15.2}$$

Here $P(j)$ and $P^k(j)$ are the fractions of amino acid of type j in the whole database and in the secondary structure of k type, respectively ($j = 1, 20; k = N_l, N, in, C, C_r$).

The *doublet characteristic* is similar to the Chou-Fasman singleton characteristic. The pair of residues can be separated by k (k may be 0, 1, 2, or 3) other residues. Preferences for a particular type of secondary structure s (α-helix, β-strand, or their N- or C-terminals) of a pair of amino acids of type i and j, separated by k residues (pair of the ijk type), are defined as:

$$D_{ijk}^s = \frac{P_{ijk}^s}{P_{ijk}} \tag{15.3}$$

Here P_{ijk} and P_{ijk}^s are the fractions of the ijk pair in the whole database and in a given secondary structure s, respectively. For N- and C-terminal regions, only those pairs that have one residue inside and one outside of these secondary structure seg-

ments were taken into consideration. The average preference of a segment to be in the s conformation (α-helix or β-strand) was calculated as the sum of all its possible pair characteristics occurring in the N-terminal, internal segment, and C-terminal of the s.

Eisenberg et al. (1984) described the *hydrophohic moment characteristic* and periodicity in hydrophobicity of a fragment of amino acid sequence as

$$M = \left[\left(\sum_{k=0}^{l-1} h_k \cos(kw)\right)^2 + \left(\sum_{k=0}^{l-1} h_k \sin(kw)\right)^2\right]^{1/2} \quad (15.4)$$

Here h_k, $k = 0, 1, \ldots, l$, is the value of the hydrophobicity (Cornette et al. 1987) of the *k-th* residue in the fragment with the length l, and w is an angle of the amino acid residue hydrophobic moment in the corresponding conformation of the polypeptide chain. The hydrophobic moments with a periodicity corresponding to α-helix ($w = 100^0$) and β-strand ($w = 180^0$) were taken as the characteristics for discriminating α-helix and β-strand segments, respectively.

The Mahalonobis distances, showing the significance of each characteristic and their combined value for each pair of alternative classes, are given in table 15.1. In all four cases (long and short α-helices and β-strands), we observe that the strongest characteristic is the average pair preference parameter, followed by the single preference parameter and hydrophobic moment. As expected, long α-helix and β-strand segments were discriminated much better than the shorter ones.

SSP Algorithm of Secondary Structure Prediction To predict the secondary structure, a nucleus of potential α-helix is searched for as a region five residues long with an average single characteristic higher than a particular threshold. The value, d, of the LDF for short α-helices is calculated for this segment. If the segment has $d < P_{as}$ (P_{as} is a threshold distinguishing short α-helices from non-helical regions), the next

Table 15.1
Mahalonobis's distances

Segment type	D	S	M	Total D^2
Long α-helices	3.97	2.31	1.77	5.19
Long β-strands	4.16	2.91	0.27	4.31
Short α-helices	2.10	1.16	0.98	2.76
Short β-strands	2.69	1.85	0.20	2.74

Significance of various characteristics of secondary structure segments measured by Mahalonobis distance D^2. D, doublet preferences; S, singleton preferences; M, hydrophobic moment.

sequence segment displaced to the right one position was examined. On condition $d > P_{as}$, the segment was expanded in both directions one residue at each step. The LDF value for a short or long α-helix for the corresponding segment was calculated depending on the segment length. A segment with the maximal d was considered a potential α-helix (the maximal extension on both sides was up to 15 positions). The search for other structures was continued after the C-terminus of the last selected helix. A similar procedure was performed for the potential β-strand segments. The result is a set of a potential α-helices and β-strands segments. In the final prediction, overlapping regions were assigned to the structures with the highest LDF value in the region of overlap. Non-overlapping edges of the segments with lower values of LDF were retained or discarded depending on their length (the minimum length for assigning an α-helix was five residues; for the β-strands, three residues).

Measures of Prediction Accuracy The jackknife procedure was used for estimating the prediction accuracy of the method on 126 nonhomologous proteins (Rost and Sander 1993), the secondary structure of which was assigned by the DSSP program (Kabsch and Sander 1983). For assessing single-residue accuracy, several performance measures were used: the percentage of correct residue predictions $Q_3 = 1/L(P_a + P_b + P_{coil}) \cdot 100\%$ and Sensitivity (Sn), Specificity (Sp), as well as Matthew's (1975) correlation coefficient (CC) as defined in chapter 9. Measures for single-residue accuracy do not completely reflect the quality of a prediction. For example, the clear wrongly predicted structure $\alpha\beta\alpha\beta\alpha\beta$ in the α-helix region would still assign correctly the conformational states of 50 percent of the residues. That is why in addition to single-residue accuracy, a measure reflecting the number of correctly predicted secondary structure segments is also introduced. In this case, α-helices and β-strands are considered as correctly identified if they had more than a pre-defined number of correctly predicted residues (Rost et al. 1994a).

The prediction accuracy Q_3 for various combinations of the considered characteristics is shown in table 15.2. Prediction using only a single characteristic gave an

Table 15.2
Single-residue prediction accuracy Q_3 for various combinations of secondary structure characteristics

Characteristics	Q_3 (%)
Singleton characteristic (S)	58.5
S + hydrophobic moment of the segment (M)	61.4
Doublet characteristic (D)	62.2
D + M	64.8
D + S + M	65.1

overall 58.2 percent prediction accuracy for three states. Therefore, the model used in this method is better than the standard Chou-Fasman method (about 50 percent accurate; Kabsch and Sander 1983), although both methods used similar parameters. Using only pair characteristics gave 62.2 percent accuracy, which is better than analogous characteristics used in the GOR III method (Gibrat et al. 1987), which have 53.5 percent accuracy. It can be seen that the addition of single characteristics only slightly increased overall prediction, from 64.8 percent to 65.1 percent. This can be explained by the high correlation between single and pair characteristics ($r = 0.80$).

Assignment of a secondary structure according to the three-dimensional coordinates performed by different methods may differ up to 21 percent (Woodcock et al. 1992). However, a detailed comparison of the secondary structure of proteins belonging to the same structural family showed that a per-residue comparison is not sufficient to assess the presence of segments in a three-dimensional structure of proteins (Rost et al. 1994).

Single-residue accuracy measures sometimes poorly reflect the actual prediction of secondary structure. For example, assigning coil state to all amino acids in the protein 4sgb gave $Q_3 = 76.7$ percent, but this protein has several missed β-structures. Conversely, for the protein 3b5c, SSP correctly predicted four out of five real α-helices and three out of five real β-strands, although Q_3 was only 56.5 percent. A simple measure for assessing the quality of predicted secondary structure segments may be the average lengths of the predicted segments (Rost and Sander 1993). The observed and predicted average lengths for α-helices are 10.6 and 10.7 amino acids, and for β-strand segments are 5.1 and 5.8 amino acids, respectively. The predicted values are very close to the real ones.

The segment prediction accuracy can be estimated by a simple measure comparable to that proposed by Taylor (1984): the structure is considered correctly predicted if it has at least two amino acids in common with the real one. It was observed that long segments were predicted much better than short ones: 89 percent of α-helices longer than eight residues and approximately 71 percent of β-strands longer than six residues were correctly located with specificity of correct prediction 0.82 and 0.78, respectively (table 15.3).

An example of SSP secondary structure prediction is given in figure 15.12. The output of the prediction program presents not only the final optimal variant of the secondary structure assignment, but also a set of potential α-helix and β-strand segments that were computed without consideration of their competition. Because the protein secondary structure is finally stabilized during the formation of the tertiary

Table 15.3
Segment prediction accuracy for short and long α-helices and β-strands

	Sn (%)	*Sp* (%)
All α-helices	75	78
Long α-helices	89	82
Short α-helices	52	51
All β-strands	51	71
Long β-strands	71	79
Short β-strands	45	64

```
Name: >gi|493758|pdb|131L|__Lysozyme_(E.C.3.2.1.17)_Mutant
ssp  Tue Sep 26 16:25:37 PDT 2000
>>gi|493758|pdb|131L|__Lysozyme_(E.C.3.2.1.17)_Mutant
 pred A:      aaaaa                            aaaaaaaaaaaaaaa
 AA           N 3.C                            N     2.3     C
 pred B:                bbbbbbb     bbbbb
 BB                     N 2.7 C     N 2.C
 Predic       aaaaa     bbbbbbb     bbbbb      aaaaaaaaaaaaaaa
 a/acid     MNIFEMLRIDEGLRLKIYKDTEGYYSIGIGHLLTKSPSLNAAKSELDKAI
                    10        20        30        40        50
 pred A:             aaaaaaaaa  aaaaaaaaaa   aaaaaaaaaaaaaaaaa
 AA                  N  5.3  C  N   3.3  C   N      4.1      C
 pred B:         bbbb                                      bbb
 BB              N2 C                                      N 1
 Predic          bbbbaaaaaaaaa  aaaaaaaaaa   aaaaaaaaaaaaaaaaa
 a/acid     GRNTNGVITKDEAEKLFNQDVDAAVRGILRNAKLKPVYDSLDAVRRAALI
                    60        70        80        90       100
 pred A:                 aaaaaaaaaa   aaaaaaaaa        aaaaaaa
 AA                      N   3.9  C   N  4.0  C        N   2.1
 pred B:    bbbb                                           bbb
 BB         .9 C                                           N 2
 Predic     bbbb         aaaaaaaaaa   aaaaaaaaa            bbb
 a/acid     NMVFQMGETGVAGFTNSLRMLQQKRWDEAAVNLAKSRWYNQTPNRAKRVI
                   110       120       130       140       150
 pred A:    aaaa
 AA            C
 pred B:    bbbb
 BB         .5 C
 Predic     bbbb
 a/acid     TTFRTGTWDAYKNL
                   160
```

Figure 15.12
An example of secondary structure prediction for Lysozyme protein. SSP outputs potential α-helices (pred A) and β-structures (pred B) with the final prediction (Predic). The last actual helix is predicted as β-strand, but we can observe potential α-helix here also with approximately the same weight.

structure, the alternative variants of the α-helix and β-strand segments may be important for methods of tertiary structure prediction. Occurrence of alternative structure may often point to wrongly predicted regions.

To improve the prediction accuracy of the method, a simple version, which treats multiple sequence alignments used as input in place of single sequences, has been developed. For testing, the alignment of homologous proteins was obtained from the HSSP database (Sander and Schneider 1991). Using the mean values of discriminant functions over the aligned sequences of homologous proteins, a prediction accuracy of 68 percent has been achieved.

The main feature of the SSP method is that it recognizes α-helix and β-strand segments rather than single residues. Another advantage is that even though the method is very simple in nature, it gives results comparable with more elaborated methods (table 15.4).

Table 15.4 lists the results of three-state accuracy for many published methods. Due to the different training sets and testing procedures, direct comparison among methods was rather difficult. Except for the PHD (Rost and Sander 1993) and SSP methods, most methods published before 1994 used databases that contained proteins with larger than 30 percent homology. Among algorithms using statistical or neural network approaches, that is, algorithms that did not use homology information, SSP, and nearest-neighbor methods (Salzberg and Cost 1992; Zhang et al. 1992; Yi and Lander 1993) give the best results. As mentioned by Rost and Sander (1993a), actual performance of some methods may be even lower, when testing on non-homologous datasets or using a jackknife procedure. For example, GOR III (Gibrat et al. 1987) was originally reported to give 63 percent accuracy, but when tested on the enlarged database gave only 56.7 percent (Geourjon and Deleage 1994).

DCS Method The method proposed by King and Sternberg (1996) is also based on linear discriminant functions involving from 10 to 27 protein features. Many of them are similar to those introduced earlier in the GOR method (Garnier et al. 1978). The features are represented as a set of *amino acid propensities* calculated for 20 residue types in three conformational states at positions $i-8$ to $i+8$, thus providing $20 \times 3 \times 17 = 1{,}020$ parameters.

Additionally, several new characteristics were introduced, such as distance from the end of a protein chain, and the moment of hydrophobicity. The distance from the chain end accounts for greater flexibility of residues at the end of protein chain. Moment of hydrophobicity evaluates possible hydrophobic patches typical for α-helices and β-structures.

Table 15.4
Comparison of prediction results

	Type	Q_3 (%)
Methods using single sequences		
GOR III (Gibrat et at. 1987)	Inform	63
Qian and Sejnowski 1988	NNw	64.3
Gibrat et al. 1987; Holley and Karplus 1989	NNw	63.2
Kneller et al. 1990		63
SM (Zhang et al. 1992)		63.5
Expert-NN (Zhang et al. 1992)	NNw	63.1
Reference network (Rost and Sander 1993b)	NNw	61.7
SSP Segment prediction	DA	65.1
Combined methods using nearest neighbors		
COMBINE (Biou et at. 1988)	NNb	65.5
Hybrid (Zhang et at. 1992)	NNb	66.4
PHD (Rost and Sander 1993, 1994)	NNw	62.6
Yi and Lander 1993	NNb	66.5
NNSSP (Salamov and Solovyev 1995)	NNb	67.6
SSPAL (Salamov and Solovyev 1997)	NNb	71.0
Methods using homology information		
Levin and Gamier 1988	NNb	63
MBR (Zhang et at. 1992)	NNb	64.5
Salzberg and Cost 1992	NNb	65.1
SSP (Solovyev and Salamov 1994)	DA	68.2
Levin et al. 1993	NNb	68.5
SDC (King and Sternberg 1996)	DA	70.1
PHD (Rost and Sander 1994)	NNw	71.6
NNSSP (Salamov and Solovyev 1995)	NNb	72.2
SSPAL (Salamov and Solovyev 1997)	NNb	73.2
PSIPRED (Jones 1999)	NNw	76–78

Single-residue prediction accuracy Q_3 for different methods of secondary structure prediction. Whenever available, the accuracy data shown for 126 nonhomologues protein data set (Rost and Sander 1993). Inform, information theory based approach; NNw, neural networks based; NNb, nearest-neighbor based; DA, discriminant analysis based approaches.

Information from aligned sequences provides additional features, such as position of deletions and insertions. The moment of conservation measures the degree of variability at a particular position.

The weak part of this approach is that the linear discriminant function per se cannot capture some higher order properties such as auto-correlation, secondary structure feedback effects, and neighborhood constraints on secondary structures. Hence a post-processing step was introduced to take this into account by using some "if-then" filtering rules. These rules are based on short patterns of five to seven consecutive positions, and they introduce corrections into secondary structure assignment made

on the previous steps. The filtering rules were found using the machine learning approach.

Overall prediction (Q_3) achieved by this algorithm when trained on 126 protein chains (the same set was used for PHD algorithm) was 70.1 percent (an overall per residue three-state accuracy).

15.4.3 Neural Networks–Based Approaches

Several algorithms using neural networks for secondary structure prediction have been developed (Rost and Sander 1993; Jones 1999). One of the most popular is the PHD algorithm by Rost and Sander (1993), which we are going to consider in detail.

The algorithm (figure 15.13) takes multiple alignments of protein sequences as an input. The sequences for the network training come from the HSSP database (Sander and Schneider 1991), where alignments of similar sequences are provided with respect to known 3D structures—that is, at least one sequence in the alignment has known structure. For prediction (when the structure is unknown) similar sequences are searched using the BLAST program (Altschul et al. 1997) and then aligned using the MaxHom algorithm (Sander and Schneider 1991).

Prediction of secondary structure makes sense for proteins, which are not homologous to any other proteins with known 3D structure; otherwise, the secondary structure can be predicted by homology with higher accuracy than by any existing secondary structure prediction algorithm (Rost 1996). To reproduce this situation in the training set, all sequences with similarity to any other sequence in the set exceeding the level given by the so-called HSSP curve (Sander and Schneider 1991) were excluded. This means exclusion of proteins, which can be aligned at 80 or more residue positions with a sequence identity ≥ 25 percent. Thus 130 protein chains were selected from seven hundred protein chains of PDB in 1992.

The multiple alignment is converted into a profile: for each position, the vector of amino acid frequencies is calculated based on the alignment. The neural net is applied sequentially to all protein sequence positions to predict secondary structure in every position as a state from three possible alternatives: helix, strand, and loop. Prediction at a given position depends on amino acid frequencies (in the profile) at that position and neighboring positions within a range defined by the window for which inputs are collected.

The architecture of the neural net (figure 15.13) consists of three layers. The first layer takes $21 \cdot w$ inputs, where 21 corresponds to 20 amino acid residues plus one for the missing residues at the beginning or at the end of the sequence; w is the window size, that is, how many neighboring residues are involved in secondary structure cal-

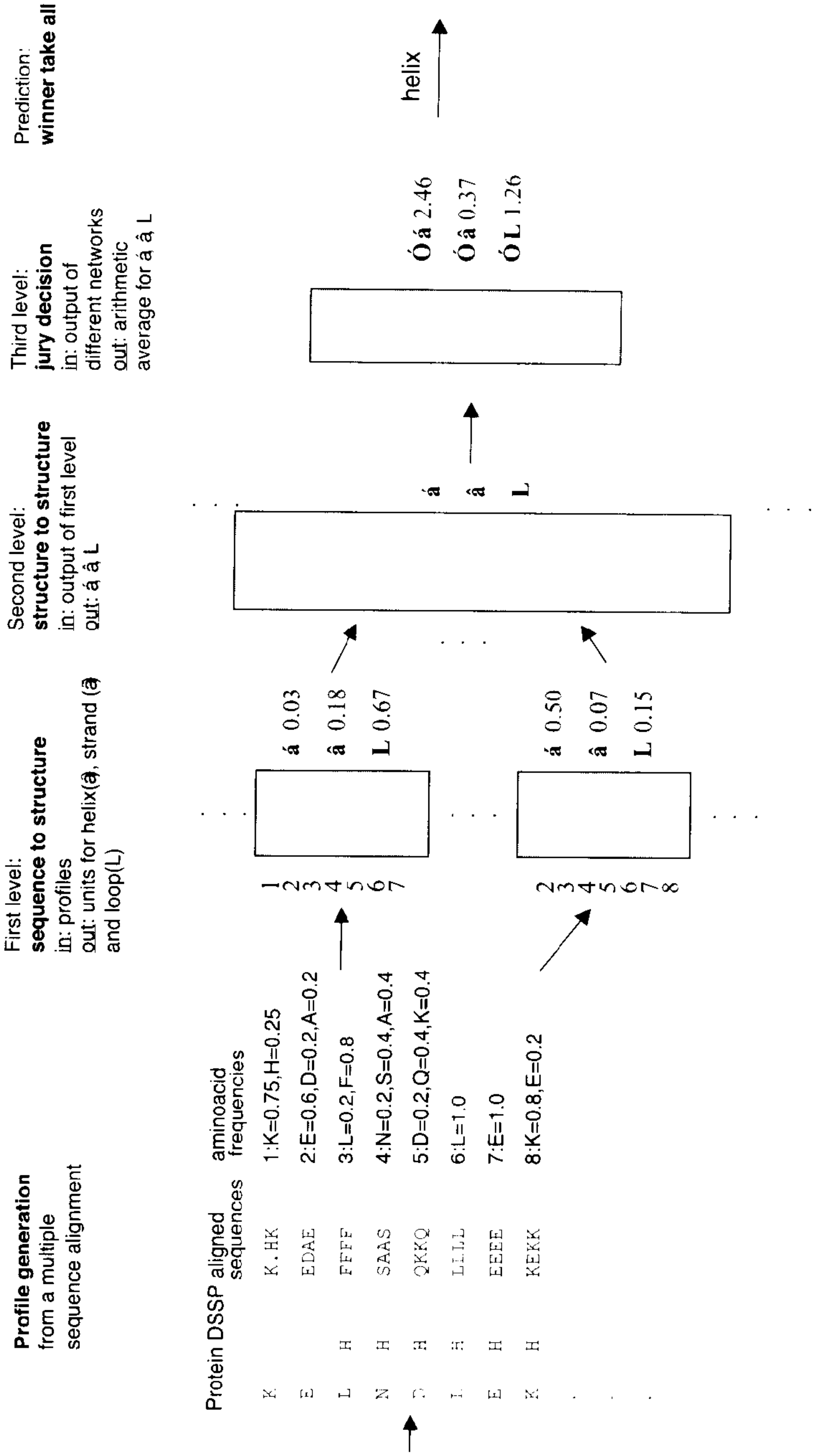


Figure 15.13
Neural-network scheme for PHD secondary structure prediction (adapted from Rost and Sander 1993).

culation for a single position—$w = 13$ was used (figure 15.13 depicts the $w = 7$ case). Amino acid frequencies for every position as well as for neighboring positions within the window are fed into the first layer of the neural network.

The frequencies are processed by the network according to

$$s_i^{2,r} = f\left(\sum_{j=1}^{N^1+1} J_{ij}^2 f\left(\sum_{k=1}^{N^0+1} J_{jk}^1 s_k^{0,r}\right)\right) \tag{15.5}$$

where J_{ij}^{λ} is the junction (describing the connection between the nodes) between unit j in layer $\lambda - 1$ and unit i in layer λ (0 for the input layer, 1 for the hidden layer, and 2 for the output layer), N^0 and N^1 are the numbers of input and hidden units, respectively, $s_k^{0,r}$ is the input from kth input unit for a given sample of data r, $f(x) = 1/(1 + e^{-\beta x})$ is the sigmoid trigger function, which describes how the output from the units is computed. Training the neural net was done using a straightforward gradient descent method. The first level produces classification of independent segments of residues with respect to the state of the central residue according to three classes of protein secondary structure: helix (α), strand (β), and loop (L). Therefore, each segment is characterized by three output values.

In the second layer of the network, the information from the first layer is modified, providing accounting of correlation between adjacent residues. Inputs are collected from the $w = 17$ units of the first layer, where w is the window size of the second layer. These inputs are converted into four binary units according to the following rule:

$$\begin{array}{llll}
0000 \text{ for frequency} & & f < 0.02 & \\
0001 \text{ for} & 0.02 \le & f < 0.33 & \\
0011 \text{ for} & 0.33 \le & f < 0.66 & (15.6) \\
0111 \text{ for} & 0.66 \le & f < 0.98 & \\
0011 \text{ for} & & f \ge 0.98 &
\end{array}$$

The output of the second layer is again three values corresponding to the three classes of protein secondary structure.

In the third layer, called "jury decision," the integration of outputs from different networks takes place. What was described above as layers one and two represents a single instance of one from many networks, which all contribute to the third level. Introduction of many networks addresses the issue of local optimums occurring in

the gradient descent training protocol. Depending on initial choices of junctions, there will be different optimal solutions achieved, referred to here as different architectures. Combining these architectures together even by the simplest way as arithmetic averages results in improved overall performance.

Overall, a three-state prediction accuracy of 70.8 percent was achieved. If we consider the prediction for only half of the sequence positions with the best reliability of prediction, then the accuracy reaches 82 percent. Further modifications of this approach increased its performance to 72 percent (Rost and Sander 1994).

The improvement was achieved by changing the algorithm to use the evolutionary information from multiple sequence alignment instead of single sequences. According to the authors, the evolutionary information is responsible for a 6–8 percent of accuracy gain compared to prediction for single sequences.

Further improvement in overall prediction accuracy was achieved by Jones (1999) in the algorithm called PSIPRED by using a slightly different neural networks design and essentially different source of aligned sequences. Instead of HSSP (used in the PHD algorithm), sequence alignments produced by the PSI-BLAST program (Altschul et al. 1997) were considered here for training and prediction (figure 15.14). PSI-BLAST also builds sequence profiles needed for the prediction algorithm. Using PSI-BLAST brings two major advances: (1) significantly faster calculation of prediction; it takes only two minutes on a Silicon Graphics Origin 200 to produce prediction starting from the query sequence; and (2) higher accuracy of prediction, about 76.5 percent compared to 72 percent for the PHD algorithm.

15.4.4 Nearest-neighbor Approaches

The basic idea of the nearest-neighbor approach is the prediction of the secondary structure state of the central residue of a given segment, based on the secondary structure of homologous segments from the proteins with known three-dimensional structure. The predicted type of secondary structure of a test residue is selected as the type of the majority of its nearest neighbors, as $max(n_\alpha, n_\beta, n_c)$, where n_α, n_β, and n_c are the numbers of nearest neighbors with the helix, strand, and coil types, respectively. The test residue is considered as the center of n consecutive amino acid residues of a sequence window. The nearest neighbors are selected by comparison of the test window sequence against all n residue windows from the database using the similarity score measure averaged over all window residues.

A key element in any nearest-neighbor prediction algorithm is the choice of a scoring measure for evaluation of similarity. The local structural environment scoring developed by Eisenberg and coworkers (Bowie et al. 1991) assigns every residue of a protein with known three-dimensional structure to an "environment class" based on

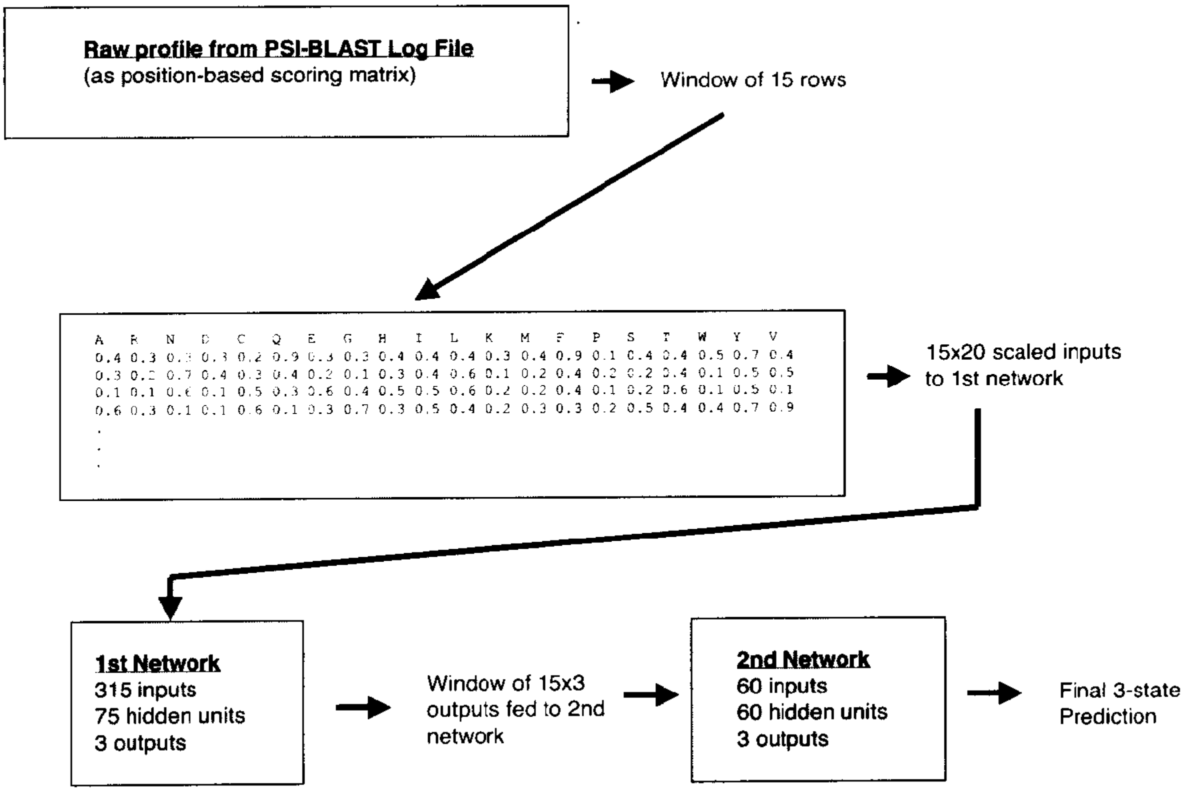


Figure 15.14
A schematic diagram of the PSIPRED method (adapted from Jones 1999).

the local structural features of the residue position, such as the solvent accessibility, polarity, and secondary structure. Yi and Lander (1993) suggested a score of matching a query residue with the database residue as the environment score plus a score, estimated by mutation matrix. The environment score (Bowie et al. 1991) for matching a residue in position i of type R_i with the database residue in position j having known local structural environment E_j is defined as

$$Env(i, j) = \log_{10}\left(\frac{P(R_i/E_j)}{P(R_i)}\right) \tag{15.7}$$

where $P(R_i \mid E_j)$ is the probability of finding residue R_i in environment E_j, and $P(R_i)$ is the probability of finding residue R_i in any environment. The score of central position i is computed as the average score in a window of length l:

$$Score(i, j) = \frac{1}{l} \sum_{k=-l/2}^{k=l/2} Env(i + k, j + k) + SM(i + k, j + k) \tag{15.8}$$

where $Env(i,j)$ is the score for matching a residue at position i of query sequence with the environment class of position j of the target protein (Salamov and Solovyev 1995) and $SM(i,j)$ is an element of an amino acid substitution matrix for residues at positions i and j of the query and target protein sequences, respectively. A substitution matrix was calculated based on the multiple sequence alignments for 126 database proteins taken from the HSSP database (Sander and Schneider 1991).

The NNSSP (Salamov and Solovyev 1995) method has created additional environment classes as N- and C-ends of α-helices and β-strands. Also, β-turns were considered as a specific class. In this way, 12 classes of secondary structure (five for α-helices: internal, N- and C-caps, the left N- and the right C-adjacent positions; the analogous five for a β-strands, β-turns, and coils) were combined with six categories of solvent accessibility/polarity, giving 72 environmental classes. Twelve classes of secondary structure were used only for nearest-neighbors selection (equation 15.8), but the only three-state secondary structure type (α, β, or c) of the center residue of nearest-neighbor windows was used for secondary structure assigning by majority rule.

An additional improvement of the predicting accuracy was reached by reducing the database where we search for amino acid fragments similar to a test protein sequence. We limited the database to a subset of proteins closest to the test protein in some general properties. Distance measure, based on the Chou-Fasman preference parameters (Chou and Fasman 1978) for helices, strands, and coils (D_{cf}), is used during selection of the subset.

$$D_{cf} = \sum_{k=1}^{3} \left((1/l_t) \sum_{j=1}^{l_t} f_t^k(j) - (1/l_b) \sum_{j=1}^{l_b} f_b^k(j) \right)^2, \qquad (15.9)$$

where $f_t(i)$ and $f_b(i)$ are frequencies of amino acid of type i; $f_t^k(j)$ and $f_b^k(j)$ are Chou-Fasman co-efficients of the amino acid residue in the j-th position for the secondary structure type k (α, β, c), and l_t and l_b are the lengths of a test and database proteins, respectively.

To exclude small elements, a few simple filtering rules were applied: (1) all helices of length 1 or 2 are converted to coils, except the case of $\beta\alpha\beta$, which is converted to $\beta\beta\beta$; (2) all strands of length 1 are converted to coils; and (3) all strands of length 2 surrounded by α-helical residues are converted to α-helices, that is, $\alpha\beta\beta\alpha$ to $\alpha\alpha\alpha\alpha$.

The NNSSP method provides about 72 percent of sustained overall three-state accuracy when we use multiple sequence alignment, or about 68 percent accuracy for single sequence input (table 15.5), when tested on a benchmark database of 126 nonhomologous proteins (Rost and Sander 1994).

Table 15.5
A comparison of prediction results using SSPAL method with results using the PHD method (Ross and Sander 1994) and the NNSSP method (Salamov and Solovyev 1995) tested on the same data set of 126 proteins (set 1) and on a big data set of 461 proteins (set 2)

	Q_3^a	Sn^a	Sp^a	C_a	$\langle L_a \rangle$	Sn^b	Sp^b	C_b	$\langle L_b \rangle$
Input: single sequences									
PHD (Set 1)	62.6	57	62	0.42	6.2	42	53	0.35	3.8
NNSSP (Set 1)	67.6	69.1	67.6	0.55	6.2	38.2	66.0	0.41	3.1
SSPAL (Set 1)	71.0	70.5	71.4	0.60	9A	52.6	66.3	0.49	4.4
SSPAL (Set 2)	71.0	72.9	71.8	0.61	9.8	51.4	67.1	0.49	4.4
Input: multiple sequence alignments									
PHD (Set 1)	71.6	70	76	0.61	9.3	62	63	0.52	5.0
NNSSP (Set 1)	72.2	72.4	76.2	0.64	11.5	52.2	67.4	0.50	4.3
SSPAL (Set 1)	73.5	75.8	73.6	0.65	9.5	52.7	72.2	0.53	4.3

Per-residue measures Q_3 residues predicted correctly in three states (helix, strand, loop) divided by all residues; Sn^a correctly predicted residues in helix divided by observed residues in helix; Sn^b the same as helix, but for strand; Sp^a correctly predicted residues in helix divided by predicted residues in helix; Sp^b the same as helix, but for strand. Ca and Cb are the Matthews correlation coefficients for helix and strand, respectively (Matthews 1975). (L_a) and (L_b) are the average segment lengths of the predicted helices and strands, respectively (for observed helices $(L_a) = 10.6$ and for observed strands $(L_b = 5.1)$.

It is more informative to provide more than one state prediction to compute probability values of three possible states for each residue. Let n_α, n_β, and n_c be numbers of the best selected nearest-neighbors for some positions of our database of proteins with known 3D structure. One can compute the proportion of α-, β-, and c-states for these positions and use these data as probability estimation (Yi and Lander 1993). We scale our n_α, n_β, and n_c values in 1–10 scale and produce a $(10 \times 10 \times 10)$ matrix with probabilities belonging to a particular state. These probabilities can be used in scoring the resemblence to a tested fold, providing a better accuracy of recognition in comparing with the one-state secondary structure prediction. An example of secondary structure prediction by NNSP approach is shown in figure 15.15.

Matching any query sequence segment with segments from the database can be viewed as un-gapped alignments between the segments of fixed length. The fixed length of segments and the absence of gaps can significantly decrease the accuracy of nearest-neighbor methods, as the best local alignments usually have different lengths for different sequence regions and often contain deletions and/or insertions. Taking this into account, several new approaches for secondary structure prediction using local or global alignments were developed (Frishman and Argos 1997; Ito et al. 1997; Salamov and Solovyev 1997).

The SSPAL method (Salamov and Solovyev 1997) used the Waterman-Eggert algorithm (1987) to compute the K best non-intersecting local alignments of a query

```
nnssp  Tue Sep 26 16:35:14 PDT 2000
>>gi|493758|pdb|131L|__Lysozyme_(E.C.3.2.1.17)_Mutant
 L=  164 SS content: a-  0.63 b=  0.05 c=  0.32
                    10        20        30        40        50
 PredSS     aaaaaaaaa     bbbb      bbb    bb     aaaaaaaaaaaa
 AA seq     MNIFEMLRIDEGLRLKIYKDTEGYYSIGIGHLLTKSPSLNAAKSELDKAI
 Prob a     88998987744222322321111111111232110128899999999998
 Prob b     00000001100014455421111467742244421000000000000000
                    60        70        80        90       100
 PredSS              aaaaaaaaaaaaaaaaaaaaa      aaaaaaaaaaaaaa
 AA seq     GRNTNGVITKDEAEKLFNQDVDAAVRGILRNAKLKPVYDSLDAVRRAALI
 Prob a     31111011188899999886889999888644232244548899999888
 Prob b     00111144200000000000000000000001112221000000000000
                   110       120       130       140       150
 PredSS     aaaaaa       aaaaaaaaaaaaaaaaaaaaaaaaa      aaaaaa
 AA seq     NMVFQMGETGVAGFTNSLRMLQQKRWDEAAVNLAKSRWYNQTPNRAKRVI
 Prob a     88888622112224557999988757799999987553311134577866
 Prob b     00111210122432100000000000000000001122111000001123
                   160
 PredSS     aaaa    aaaaaa
 AA seq     TTFRTGTWDAYKNL
 Prob a     55641113577877
 Prob b     33221011111011
```

Figure 15.15
An example of NNNSP secondary structure prediction for Lysozime protein. NNSPP computes probability of β- and β-structures for each position. The small last α-helix actually is G-helix according to DSSP and we can observe that it has lower probability values for α-helix than the other predicted α-helices.

sequence with each sequence of the subset selected from the database of known 3D structures. Then the prediction is based on the information about secondary structure states of aligned sequence segments. The term "non-intersecting" assumes that each next suboptimal alignment do not share any of the aligned pairs with the preceding alignments (Waterman and Eggert 1987). It was shown that if we choose K large enough (in the range of 30–60), it significantly improves the prediction accuracy. For a given query protein, the approach produced up to $N \times K$ local alignments, variously located along the entire sequence, where N is the number of used database sequences. The score of a local alignment is taken as the score assigned to a given aligned position. The score of each one of three conformational states is computed as the sum of alignment scores with corresponding positions belonging to a particular conformational state. The first 50 alignments with the highest scores were taken into account. The predicted state of position is the state having the highest total score. The best performance ($Q_3 = 71.2\%$) was observed at $K = 30$ when the gap-opening penalty equals -20 and the gap-extension penalty equals -10. Another test of the method was performed using a representative list of 461 proteins (PDB_SELECT)

Table 15.6
Percentage of OVER, UNDER, and WRONG SSPAL predictions on the data set of 126 proteins

Input	OVER	UNDER	WRONG
Single sequences	10.5	15.3	3.3
Multiple sequence alignments	9.4	14.4	2.6

with less than 25 percent sequence similarity (Hobohm and Sander 1994). Four hundred and sixty-one protein sequences were selected from 486 protein chains of this list (PDB_SELECT, March 1996), excluding all PDB entries with only known C^{α} atom coordinates and membrane proteins according to the SCOP (Murzin et al. 1994). For this dataset, a prediction accuracy of 71 percent was achieved. In addition to the Q_3 score, we also computed several other measures of prediction accuracy (table 15.5) that take into account some of the characteristics of predictions that can be important for building tertiary structural models.

Secondary structure predictions can be evaluated by subcategorizing the incorrect predictions into three categories: OVER (extra α or β), UNDER (unpredicted α or β), and WRONG (α as β or β as α) (Defay and Cohen 1995). Table 15.6 presents the values of these categories for SSPAL method. It is assumed that the WRONG predictions can affect the fold recognition more seriously.

The SSPAL method presents an improvement of nearest-neighbor algorithms using local alignments and their scores instead of fixed-length un-gapped segment pairs. It is simple in realization and shows a certain increase in accuracy with using the database constructed from PDB_SELECT list with a 35 percent cutoff value. An additional advantage of the method is obviously a high level of prediction accuracy (up to 100 percent) when we analyze a sequence having some similarity with one of the database sequences.

In the previous realization of nearest-neighbor methods, the score was computed as the score of similarity of two short segments and the best nearest-neighbor score was not significantly higher than the scores of the other nearest-neighbors. In the current method, the prediction is mostly based on the first optimal alignment between a query sequence and homologous target, because the score of this alignment will significantly dominate the scores of all other alignments (figure 15.16).

There are several attempts to create a consensus method of secondary structure prediction by combining the prediction from different approaches (Viswanadhan et al. 1991; Zhang et al. 1992). Jpred (combining the NNSSP, DSC, PREDATOR, MULPRED, ZPRED, and PHD methods) provides better and probably more stable results then each of the single methods used (Cuff et al. 1998; Cuff and Burton 1999).

```
a) 50 local alignments:
                    243
                     ...GEFDIDCDNLSYMPTVVFEINGKMYPLTPSAYTSQDQGFCTS
                     ...ccbbbcccccccccccbbbbbccbbbbbchhhhbbbbccbbbb
                                                  *
5er2E 3              ...GGYVGPCSA--TLPSFTFGVGSARIVIPGDYFGPISTGSC
K=1 S=3267             cbbbbbccc--cccbbbbbccbbbbbchhhhbbccccbbb
                                                  *
4        4rhv1 197       YGITVLNHMGSMAFRIVNE
K=2     k=2 S=384        ccccccccccbbbbbbccc
                                                  *
             3cd4 1                 KKVVLGKKGDTVELTCTA
             k=3 S=374              cbbbbbbccccbbbbcccc
                                                  *
                                                 ...
                   6cpp 170        YLTDQMT
                    k=50 s=50      hhhhhhh
                                                  *
b) 50 nearest-neighbor segments

                    243
                     ...GEFDIDCDNLSYMPTVVFEINGKMYPLTPSAYTSQDQGFCTS
                     ...ccbbbcccccccccccbbbbbccbbbbbchhhhbbbbccbbbb
                                                  *
                     4rhv1 202 LNHMGSMAFRIVNEHDE
                     K=1 S=304 cccccbbbbbbbccbbbb
                                                  *
                     5hvpa 5   LWQRPLVTIKIGGQLKE
                     K=2 S=238 cccccbbbbbbccbbbb
                                                  *
                     6cts 178  IAKLPCVAAKIYRNLYR
                     K=3 S=234 hhhhhhhhhhhhhhhhc
                                                  *
                                                 ...
                     4pfk 296  IAEALANKHTIDQRMYA
                     k=50 S=44 hhhhcccccccchhhhh
                                                  *
```

Figure 15.16
Prediction of the secondary structure state of Phe at position 261 of chymosin B (4 cm) based on: (a) the 50 best local alignments; and (b) the 50 nearest-neighbor segments. The PDB identifier, the first position of matched segment and the score are shown.

Table 15.7
Web servers for secondary structure prediction

Name of program	WWW address
DSC	http://www-bioweb.pasteur.fr/seqanal/interfaces/dsc-simple.html http://www.aber.ac.uk/~phiwww/prof/
Jpred (consensus method)	http://jura.ebi.ac.uk:8888/
PSIPRED	http://insulin.brunel.ac.uk/psipred/
PHD	http://cubic.bioc.columbia.edu/predictprotein/predictprotein.html http://www.ebi.ac.uk/~rost/predictprotein/
PREDATOR	http://www.embl-heidelberg.de/argos/predator/run_predator.html
SSPAL, NNSSP, SSP	http://dot.imgen.bcm.tmc.edu:9331/psspredіction/pssp.html http://www.softberry.com/protein.html

Nearest-neighbor approaches have the potential to improve their performance from about 70–75 percent to 80 percent with an increasing number of known tertiary structures and the number of homologous sequences in protein databases, as we observed recently for neuralnetwork approaches (Jones 1999; Petersen et al. 2000). Further progress will probably be difficult due to dependence of small secondary structure elements formation on the 3D structural environments, as well as on the limitations of secondary structure assignment. However, we can envision some improvement in secondary structure assignment and further development of approaches that predict the entire secondary structure segments (Solovyev and Salamov 1994; Schmidler et al. 2000; Petersen et al. 2000) rather than single residue state.

Prediction of secondary structure by the methods described above is available via the World Wide Web (table 15.7). Protein secondary structure assignment from atomic coordinates by the STRIDE (Frishman and Argos 1995) program can be done at http://bioweb.pasteur.fr/seqanal/interfaces/stride-simple.html.

References

Afifi, A. A., and Azen, S. P. (1979). *Statistical Analysis: A Computer-oriented Approach.* New York: Academic Press.

Altschul, S. F., Madden, T. L., Schäffer, A. A., Zhang, J., Zhang, Z., Miller, W., and Lipman, D. J. (1997). Gapped BLAST and PSI-BLAST: A new generation of protein database search programs. *Nucl. Acids Res.* 25: 3389–3402.

Bowie, J. U., Luthy, R., and Eisenberg, D. (1991). A method to identify protein sequences that fold into a known three-dimensional structure. *Science* 253: 164–170.

Chothia, C. (1976). The nature of the accessible and buried surfaces in proteins. *J. Mol. Biol.* 105: 1–12.

Chou, P. Y., and Fasman, G. D. (1974a). Conformational parameters for amino acids in helical, beta-sheet, and random coil regions calculated from proteins. *Biochemistry* 13(2): 211–222.

Chou, P. Y., and Fasman, G. D. (1974b). Prediction of protein conformation. *Biochemistry* 13(2): 222–245.

Chou, P. Y., and Fasman, G. D. (1977). Beta-turns in proteins. *J. Mol. Biol.* 115: 135–175.

Chou, P., and Fasman, G. (1978). Empirical predictions of protein conformation. *Annu. Rev. Biochem* 47: 251–276.

Cohen, F. E., Abarbanel, R. M., Kuntz, I. D., and Fletterick, R. J. (1983). Secondary structure assignment for alpha/beta proteins by a combinatorial approach. *Biochemistry* 22(21): 4894–4904.

Cohen, F. E., Abarbanel, R. M., Kuntz, I. D., and Fletterick, R. J. (1986). Turn prediction in proteins using a pattern-matching approach. *Biochemistry* 25(1): 266 275.

Cornette, J. L., Cease, K. B., Margalit, H., Spouse, J. L., Berzolsky, J. A., and DeLisi, C. (1987). Hydrophobicity scales and computational techniques for detecting amphipathic structures in proteins. *J. Mol. Biol.* 195: 659–685.

Cuff, J. A., and Barton, G. J. (1999). Evaluation and improvement of multiple sequence methods for protein secondary structure prediction, *Proteins: Struct. funct. genet.* 34: 508–519.

Cuff, J. A., Clamp, M. E., Siddiqui, A. S., Finlay, M., Barton, G. J. (1998). Jpred: A consensus secondary structure prediction server. *Bioinformatics* 14: 892–893.

Defay T., and Cohen, F. E. (1995). Evaluation of current techniques for ab initio protein structure prediction. *Proteins* 23(3): 431–445.

Di Francesco, V., Munson, P. J., and Garnier, J. (1999). FORESST: Fold recognition from secondary structure predictions of proteins. *Bioinformatics* 15(2): 131–140.

Eisenberg, D., Weiss, R. M., and Terwilliger, T. C. (1984). The hydrophobic moment detects periodicity in protein hydrophobicity. *Proc. Natl. Acad. Sci. USA* 81: 140–144.

Fischer, D. (2000). Hybrid fold recognition: Combining sequence derived properties with evolutionary information. *Proceedings of Pac. Symp. Biocomput.* 119–130. World Scientific Publishing.

Fischer, D., and Eisenberg, D. (1996). Fold recognition using sequence-derived properties. *Prot. Sci.* 5: 947–955.

Frishman, D., and Argos, P. (1995). Knowledge-based protein secondary structure assignment. *Proteins* 23(4): 566–579.

Frishman, D., and Argos, P. (1997). Seventy-five percent accuracy in protein secondary structure prediction. *Proteins* 27: 329–335.

Garnier, J., Osguthorpe, D., and Robson, B. (1978). Analysis of the accuracy and implications of simple methods for predicting the secondary structure of globular proteins. *J. Mol. Biol.* 120: 97–120.

Gibrat, J. F., Garnier, J., and Robson, B. (1987). Further developments of protein secondary structure prediction using information theory. New parameters and consideration of residue pairs. *J. Mol. Biol.* 198: 425–443.

Geourjon, C., and Deleage, G. (1994). SOPM: A self-optimized method for protein secondary structure prediction. *Protein Eng.* 7: 157–164.

Hargbo, J., and Elofsson, A. (1999). Hidden Markov models that use predicted secondary structures for fold recognition. *Proteins* 36(1): 68–76.

Hobohm, U., and Sander, C. (1994). Enlarged representative set of protein structures. *Prot. Sci.* 3: 522–524.

Holley, H. L., and Karplus, M. (1989). Protein secondary structure prediction with a neural network. *Proc. Natl. Acad. Sci. USA* 86: 152–156.

Ito, M., Matsuo, Y., and Nishikawa, K. (1997). Prediction of protein secondary structure using the 3D-1D compatibility algorithm. *Comput. Appl. Biosci.* 13(4): 415–424.

Jones, D. T. (1999). Protein secondary structure prediction based on position-specific scoring matrices. *J. Mol. Biol.* 292(2): 195–202.

Kabsch, W., and Sander, C. (1983). Dictionary of protein secondary structure: Pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers* 22: 2577–2637.

King, R. D., Saqi, M., Sayle, R., and Sternberg, M. J. (1997). DSC: Public domain protein secondary structure predication. *Comput Appl Biosci.* 13(4): 473–474.

King, R. D., and Sternberg, M. J. E. (1996). Identification and application of the concepts important for accurate and reliable protein secondary structure prediction. *Prot. Sci.* 5: 2298–2231.

Koretke, K. K., Russell, R. B., Copley, R. R., and Lupas, A. N. (1999). Fold recognition using sequence and secondary structure information. *Proteins* 37: 141–148.

Levitt, M. (1978). Conformational preferences of amino acids in globular proteins. *Biochemistry* 17: 4277 4285.

Levitt, M., and Greer, J. (1977). Automatic identification of secondary structure in globular proteins. *J. Mol. Biol.* 114: 181–239.

Lim, V. I. (1974). Algorithms for prediction of α-helices and β-structural regions in globular proteins. *J. Mol. Biol.* 88: 873–894.

Linderstrom-Lang, K. U., and Schnellman, J. A. (1959). Protein structure and enzyme activity. In *The Enzymes*, Boyer, P. D., ed. New York: Academic Press. 443–510.

Matthews, B. W. (1975). Comparison of the predicted and observed secondary structure of T4 phage lysozyme. *Biochim. Biophys. Acta* 405: 442–451.

Murzin, A. G., Lesk, A. M., and Chothia, C. (1994). Principles deterniming the structure of beta-sheet barrels in proteins. II. The observed structures. *J. Mol. Biol.* 236: 1382–1400.

Nagano, K. (1973). Logical analysis of the mechanism of protein folding. I. Predictions of helices, loops and beta-structures from primary structure. *J. Mol. Biol.* 75: 401–420.

Némethy, G., and Scheraga, H. A. (1977). Protein folding. *Quarterly Reviews of Biophysics* 10: 239 252.

Petersen, T. N., Lundegaard, C., Nielsen, M., Bohr, H., Bohr, J., Brunak, S., Gippert, G. P., and Lund, O. (2000). Prediction of protein secondary structure at 80% accuracy. *Proteins* 41(1): 17 20.

Presta, L. G., and Rose, G. D. (1988). Helix signal in proteins. *Science* 240: 1632–1641.

Ptitsyn, O. B., and Finkelstein, A. V. (1983). Theory of protein secondary structure and algorithm of its prediction. *Biopolymers* 22: 15–22.

Qian, N., and Sejnowski, T. J. (1988). Predicting the secondary structure of globular proteins using neural network models. *J. Mal. Dial.* 202: 865–884.

Palau, J., and Puigdoménech, P. (1974). The structural code for proteins: Zonal distribution of amino acid residues and stabilization of helices by hydrophobic triplets. *J. Mol. Biol.* 88: 457–469.

Pauling, L., Corey, R. B., and Branson, H. R. (1951). Two hydrogen-bonded helical configurations of the polypeptide chain. *Proc. Natl. Acad. USA* 37: 729.

Ramachandran, G. N., Venkatachalam, C. M., and Krimm, S. (1966). Stereochemical criteria for polypeptide and protein chain conformations. 3. Helical and hydrogen-bonded polypeptide chains. *Biophysical Journal* 6: 849–872.

Richardson, J. S. (1981). The anatomy and taxonomy of protein structure. *Advances in Protein Chemistry* 34: 167–339.

Richardson, J. S., and Richardson, D. C. (1988). Amino acid preferences for specific locations at the ends of α-helices. *Science* 240: 1648–1652.

Rost, B. (1996). PHD: Predicting one-dimensional protein structure by profile-based neural networks. In *Methods in Enzymology*, Doolittle, F. ed. New York: Academic Press, 525–539.

Rost, B., and Sander, C. (1993). Prediction of protein secondary structure at better than 70% accuracy. *J. Mol. Biol.* 232: 584–599.

Rost, B., and Sander, C. (1994). Combining evolutionary information and neural networks to predict protein secondary structure. *Proteins: Struc. Funct. Genet.* 19: 55–72.

Rost, B., Sander, C., and Schneider, R. (1994). Redefining the goals of protein secondary structure prediction. *J. Mol. Biol.* 235: 13–26.

Russel, R. B., Copley, R. R., and Barton, G. J. (1996). Protein fold recognition by mapping predicted secondary structures. *J. Mol. Biol.* 259: 349–365.

Salamov, A. A., and Solovyev, V. V. (1995). Prediction of protein secondary structure by combining nearest-neighbor algorithms and multiple sequence alignments. *J. Mol. Biol.* 247: 11–15.

Salamov, A. A., and Solovyev, V. V. (1997). Protein secondary structure prediction using local alignments. *J. Mol. Biol.* 268: 31–36.

Salzberg, S., and Cost, S. (1992). Predicting protein secondary structure with nearest-neighbor algorithm. *J. Mol. Biol.* 227: 371–374.

Sander, C., and Schneider, R. (1991). Database of homology-derived dstructures and the structural meaning of sequence alignment. *Proteins: Struct. Funct. Genet.* 9: 56–68.

Schmidler, S. C., Liu, J. S., and Brutlag, D. L. (2000). Bayesian segmentation of protein secondary structure. *J. Comput. Biol.* 7(1–2): 233–248.

Schulz, G., and Schirmer, R. (1979). Principles of Protein. New York: Springer-Verlag.

Solovyev, V. V., and Salamov, A. A. (1991). Method of calculation of discrete secondary structures in globular proteins. *Mol. Biol.* 25(3): 810–824.

Solovyev, V. V., and Salamov, A. A. (1994). Predicting alpha-helix and beta-strand segments of globular proteins. *Comput. Appl. Biosci.* 10(6): 661–669.

Taylor, W. R. (1984). An algorithm to compare secondary structure predictions. *J. Mal. Dial.* 173: 512–521.

Viswanadhan, V. N., Denckla, B., and Weinstein, J. N. (1991). New joint prediction algorithm (Q7-JASEP) improves the prediction of protein secondary structure. *Biochemistry* 30(46): 11164–11172.

Waterman, M. S., and Eggert, M. (1987). A new algorithm for best subsequence alignments with application to tRNA-rRNA comparisons. *J. Mol. Biol.* 197: 723–728.

Woodcock, S., Mornan, J. P., and Henrissat, B. (1992). Detection of secondary structure elements in proteins by hydrophobic cluster analysis. *Prot. Engin.* 5: 629–635.

Yi, T.-M., and Lander, E. S. (1993). Protein secondary structure prediction using nearest-neighbor methods. *J. Mol. Biol.* 232: 1117–1129.

Zhang, X., Mesirov, J. P., and Waltz, D. L. (1992). Hybrid system for protein secondary structure prediction. *J. Mol. Biol.* 225: 1049–1063.

Zvelebil, M. J., Barton, G. J., Taylor, W. R., and Sternberg, M. J. (1987). Prediction of protein secondary structure and active sites using the alignment of homologous sequences. *J. Mol. Biol.* 195(4): 957–961.

16 Computational Methods for Protein Folding: Scaling a Hierarchy of Complexities

Hue Sun Chan, Hüseyin Kaya, and Seishi Shimizu

A knowledge of the molecular basis of life is crucial for advances in biomedical and agricultural research. Proteins are a diverse class of biomolecules performing vital functions in all living things. Therefore, developing a fundamental understanding of how proteins fold is of immense intellectual and technological significance. A solution to the protein folding problem is often likened to the deciphering of the "second genetic code" (Chan and Dill 1993, and references therein). The question is simple to pose: Given a specific sequence of amino acids, how do physical and chemical forces determine its myriad properties, especially the essentially unique folded structure of a globular protein? In principle, this question should be answerable because numerous proteins fold and unfold reversibly in vitro. Protein folding is much more complex in vivo, with the involvement of chaperones and other cellular machineries. Nevertheless, any first-principles account of these complicated processes must begin with an understanding of the considerably simpler folding processes in vitro.

16.1 Physics and Knowledge-Based Approaches to Protein Folding

Like all natural phenomena, protein folding is a physico-chemical process. Hence, all methods for protein structure prediction require various degrees of understanding of, or assumption about the underlying energetics. Recent protein structure prediction techniques, which include comparative or homology modeling, fold recognition, and "ab initio" approaches, have focused primarily on empirical and statistical analyses of sequence and structure databases. Because of their *inductive* nature, techniques that rely predominantly on correlative parameters derived from collections of existing structural and sequence information are called "knowledge-based." These methods are often contrasted with "physics-based" methods that attempt to *deductively* explain or predict protein behaviors from elementary physical forces. At present, knowledge-based approaches—some of which are covered in excellent chapters elsewhere in this volume—seem to be more successful than any current physics-based technique in predicting native structures from given amino acid sequences. In fact, as Moult et al. (1999) and Shortle (2000) have noted, most current protein structure prediction algorithms give little consideration to biophysical questions such as balance of forces, energetic components, or folding pathways. Indeed, with the advent of experimental structural genomics (see, e.g., Šali and Kuriyan 1999), comparative modeling will certainly become increasingly important for providing structural and functional insight into the world's rapidly expanding sequence databases.

Protein folding is complex. To make progress, correlations among observations of all kinds need to be sought at every level, even if the underlying physical reasons are not quite known. Knowledge-based methods for protein structure prediction have contributed tremendously in this regard. But one should never lose sight of the importance of developing physics-based theories. First, aside from their intrinsic intellectual value, such endeavors are necessary for justifying or improving knowledge-based methods (Osguthorpe 2000); examples are discussed below. Second, even if technologies for predicting protein native structure turn out to be achievable by purely knowledge-based means, to ascertain how a protein functions or malfunctions often requires biophysical information that cannot be gleaned from a static picture of its native conformation alone. Notably, a protein's dynamic properties can be crucial for its functions (Kay 1998; Zidek et al. 1999; Forman-Kay 1999). Conformational distribution and fluctuation in a protein's denatured state (Wu 1931; Dill 1990; Edsall 1995; Shortle 1996) are important for native thermodynamic stability, and bear directly on a protein's tendency to adopt disease-causing misfolded and/or aggregated forms (Cohen and Prusiner 1998; Kelley 1998; Dobson 1999). Physics-based theories, which are the main focus of this chapter, are necessary to address these essential aspects of protein behavior.

Our purpose here is to provide a broad-stroke panoramic view of this area of research. As illustrations of general principles, several physics-based theoretical and computational approaches to protein folding are highlighted. We place special emphasis on critical evaluations of methods and identifying unresolved issues that are crucial for future progress. We assess the strengths and limitations of a number of approaches, focusing especially on features and assumptions that are important in determining whether a model is proteinlike or not, but whose ramifications have not been fully appreciated in the literature. The following is an outline of the rest of this chapter: (1) a rough sketch of all-atom simulations and driving forces in protein folding; (2) an introductory discussion of solvation effects in protein energetics and the complexities they entail, using electrostatic and hydrophobic interactions as examples; (3) a delineation of the advantages and limitations of using simplified chain representations to study protein folding, and the physical implications of popular modeling approaches; and (4) summaries of recent applications of simple lattice protein models to evolutionary landscapes, protein aggregation, protein calorimetric cooperativity, and folding kinetics.

16.2 Driving Forces in Protein Folding and Model Potentials

16.2.1 All-Atom Models

A protein molecule is a large collection of covalently linked atoms. In principle, its properties should be deducible from these atoms' interactions among themselves and with the atoms of the solvent molecules. Computational approaches using all-atom empirical force fields are predicated on this premise. Typically, the potential energy (force field) V in these studies is a sum of contributions and a function of the spatial position vectors $\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_N$ of all atoms in the system (Leach 1996; Rapaport 1997). For example, equation 3.1 in Leach (1996) may be rewritten to make explicit its dependences on atomic coordinates:

$$
\begin{aligned}
V(\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_N) \\
= \sum_{[i<j]}^{N} \frac{w_{ij}}{2} (|\mathbf{r}_i - \mathbf{r}_j| - b_{ij}^0)^2 + \sum_{\substack{[i,j] \\ [j,k] \\ i<k}}^{N} \frac{v_{ijk}}{2} \left(\cos^{-1} \frac{(\mathbf{r}_i - \mathbf{r}_j) \cdot (\mathbf{r}_j - \mathbf{r}_k)}{|\mathbf{r}_i - \mathbf{r}_j| \, |\mathbf{r}_j - \mathbf{r}_k|} - \theta_{ijk}^0 \right)^2 \\
+ \sum_{\substack{[i,j] \\ [j<k] \\ [k,l]}}^{N} \sum_{n=0}^{m} \frac{V_{ijkl}^{(n)}}{2} \left[1 + \cos\left(n \cos^{-1} \frac{[(\mathbf{r}_i - \mathbf{r}_j) \times (\mathbf{r}_j - \mathbf{r}_k)] \cdot [(\mathbf{r}_j - \mathbf{r}_k) \times (\mathbf{r}_k - \mathbf{r}_l)]}{|(\mathbf{r}_i - \mathbf{r}_j) \times (\mathbf{r}_j - \mathbf{r}_k)| \, |(\mathbf{r}_j - \mathbf{r}_k) \times (\mathbf{r}_k - \mathbf{r}_l)|} - \gamma_{ijkl} \right) \right] \\
+ \sum_{\substack{i<j \\ \text{nonbonded}}}^{N} \left\{ 4\mathcal{E}_{ij} \left[\left(\frac{\sigma_{ij}}{|\mathbf{r}_i - \mathbf{r}_j|} \right)^{12} - \left(\frac{\sigma_{ij}}{|\mathbf{r}_i - \mathbf{r}_j|} \right)^{6} \right] + \frac{q_i q_j}{4\pi\epsilon_0 |\mathbf{r}_i - \mathbf{r}_j|} \right\} \qquad (16.1)
\end{aligned}
$$

where the square brackets in the expressions $[i, j]$ and $[i < j]$ indicate that a summation is only over atoms i and j that are covalently linked (to form bond i–j); w_{ij} and v_{ijk} are force constants for deviations of bond length (between atoms i and j) and bond angle (subtended by bonds i–j and j–k) from their reference (natural) values b_{ij}^0 and θ_{ijk}^0, respectively; $V_{ijkl}^{(n)}$'s are "barrier-height" parameters and γ_{ijkl} is the phase factor for the torsion angle between bonds k–l and i–j; $\mathcal{E}_{ij}$ and σ_{ij} are, respectively, the well depth and collision diameter of Lennard-Jones interactions between atoms i and j, q_i is the effective electric charge of atom i, ϵ_0 is vacuum permittivity,[1] and the

1. Electrostatic relations are given in SI units. Equivalent formulas in Gaussian units may be obtained by replacing $1/\varepsilon_0$ with 4π in Eq. (16.1) and subsequent expressions.

last summation over pairwise non-bonded interactions is over atom pairs i, j that are not covalently bonded.

Because of the large number of atoms involved, simulations of proteins using all-atom force fields such as equation (16.1) are computationally intensive. The CPU time needed is many orders of magnitude that of the physical time simulated. For an up-to-date review of advances in relatively long-timescale protein and peptide simulations, see Daggett 2000. To date, the longest simulation of any protein folding process has been Duan and Kollman's 1998 study of the 36-residue villin headpiece (involving 9,295 atoms), tracking a single dynamic trajectory that corresponds to 1 microsecond of physical time. Results from some recent all-atom simulations are encouraging. For example, a short (7-unit) β-peptide in methanol has been successfully folded by molecular dynamics simulation to its experimentally determined stable conformation (Daura et al. 1998). However, no corresponding success has yet been reported for globular proteins. A reason may be that even the fastest folding proteins known to date take tens of microseconds to fold (see, e.g., Spector and Raleigh 1999), which is more than one order of magnitude longer than the longest simulations performed so far.

The functional form in equation (16.1) and others similar to it represent the most detailed atomistic considerations currently feasible for building workable protein folding models. However, as a matter of principle, it is important to realize that they are still drastic simplifications of the real physics. In particular, the pairwise potentials in the last summation do not model the true fundamental interactions between pairs of isolated atoms. Instead, they are empirically parameterized to approximate the effects of many-body interactions that are often intrinsically not pairwise additive. A clear example is that the dipole moment of many water models are significantly stronger than that of a single isolated water molecule in the gas phase, because the models are parameterized to reproduce properties of liquid water. It follows that parameters in empirical force fields are not universal, as they need to be optimized for particular sets of applications (Leach 1996). Indeed, even for the relatively simple system of argon atoms in the liquid state, quantum mechanical calculations have shown that there is no universal effective pairwise potential that can adequately reproduce three-body dispersion (London) interactions for all applications (van der Hoef and Madden 1999).

16.2.2 Rationale for Coarse-Grained Statistical Mechanics Models

All-atom treatments are impractical if broad conformational sampling is required because they are computationally intensive, and general inferences cannot be reliably drawn from a small number of molecular dynamics trajectories (Braxenthaler et al.

1997). In view of these limitations and the non-universality of empirical force-field parameters, researchers have also pursued complementary coarse-grained statistical mechanical models of proteins. As for ferromagnetism in condensed matter physics, a hierarchy of organizing principles is expected to be needed to bridge our conceptual understanding of physical phenomena of atomic and macroscopic length scales as well as phenomena that take place at intermediate "mesoscopic" length scales to which proteins belong (Laughlin et al. 2000). It is natural to use coarse-grained models for low-resolution descriptions in this regard. Furthermore, coarse-grained models are computationally more tractable, and, because of their relative simplicity, can often provide insight that would have been obscured otherwise. A case in point is the recent "Gaussian network model" of collective motions and correlations in folded states of globular proteins. At a computational cost approximately three orders of magnitude less than that required by all-atom calculations, the predicted dynamics of these coarse-grained models are similar to that obtained by all-atom molecular dynamics simulation and normal mode analysis (Bahar et al. 1999; Haliloglu and Bahar 1999; Doruker et al. 2000).

In "big-picture" conceptual formulations of protein folding and in coarse-grained models, it is customary to classify noncovalent interactions into a few energetic components or interaction types, namely hydrophobic, hydrogen bonding, electrostatic, and dispersion (attractive) and repulsive van der Waals forces (reviewed by Kauzmann 1959; Dill 1990; Honig and Yang 1995). In terms of the atomic interactions presumed by equation 16.1 above, "hydrophobic interaction" may be viewed as the combined effect of van der Waals interactions between nonpolar atoms in the protein and their interactions with the surrounding water and cosolvent molecules. Hydrogen bonding interaction is sometimes treated as a type of electrostatic interaction, as provided by the last term in equation 16.1 or sometimes it is implemented by extra terms (Leach 1996). As Cooper (1999) noted, the short list of interaction types that are believed to be relevant to protein folding has not changed for the past 40 years. However, even at this coarse-grained level, a coherent physical picture is still lacking. For instance, even the basic question of whether hydrogen bonding stabilizes or destabilizes native states of proteins remains controversial (Myers and Pace 1996; Ben-Tal et al. 1997; Pace et al. 1998).

16.3 Solvation Effects and Non-additivity of Potentials of Mean Force

A major difficulty in accounting for protein energetics is the effect of the aqueous solvent (Roux and Simonson 1999). Solvent-mediated interactions between constitu-

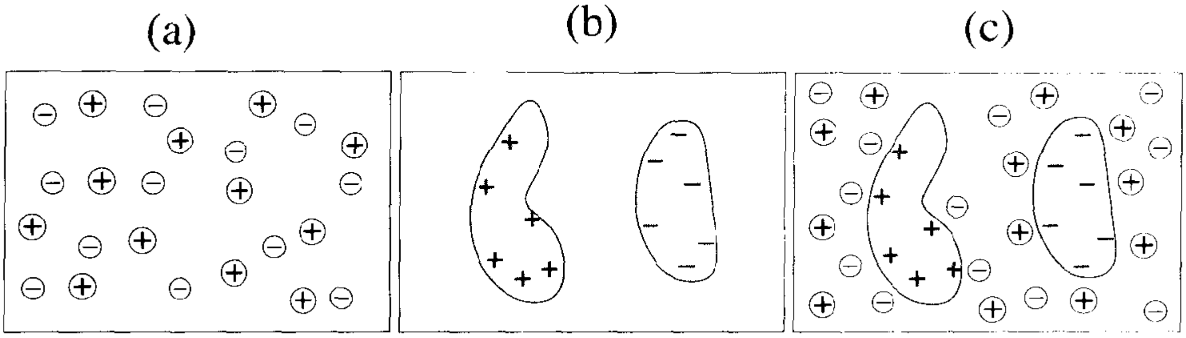


Figure 16.1
Schematics of solvent-mediated electrostatic interactions. (a) Solvent with dissolved ions. (b) Direct interaction in vacuum between two macromolecules (shaded shapes). (c) Solvent-mediated interaction between the two macromolecules. Ions in the solvent tend to migrate near oppositely charged macromolecules, partially shielding their direct electrostatic interaction.

ent groups of a protein can have more complex properties than the corresponding direct interactions in vacuum. For instance, ions in a solvent can lead to partial screening of the direct electrostatic interactions between two macromolecules (figure 16.1), and hydrophobic effects in proteins involve protein-water as well as water-water interactions (figure 16.2).

16.3.1 Example: Electrostatic Interactions

To illustrate the complexities of solvation effects, we present a brief outline of the Poisson-Boltzmann approach, which is an approximate continuum treatment of solvent-mediated electrostatic interactions. The analysis starts with the Poisson equation, $\nabla \cdot [\epsilon(\mathbf{r})\nabla\phi(\mathbf{r})] = -\rho(\mathbf{r})/\epsilon_0$, which is a re-statement of Gauss' law generalized to include polarizable media. Here $\epsilon(\mathbf{r})$ is the position-dependent dielectric constant of a given system, $\rho(\mathbf{r})$ is the free charge density (which excludes the bound charges, whose effects are approximately accounted for by $\epsilon(\mathbf{r})$ of the dielectric media), and $\phi(\mathbf{r})$ is the electrostatic potential for the corresponding *macroscopic* electric field (Jackson 1975).

In the presence of mobile ions in the solvent and free charges fixed at a given set of spatial positions (figure 16.1c), $\rho(\mathbf{r})$ may be expressed as a sum of fixed charge density $\rho_{\text{fix}}(\mathbf{r})$ and contributions from the mobile ions. Let the discrete charges of the ions be $+q$ (> 0) and $-q$. The standard Poisson-Boltzmann argument posits that the equilibrium populations of the $\pm q$ mobile ions at any position $\mathbf{r}$ accessible to the ions are proportional to the Boltzmann factors $\exp(\mp q\phi(\mathbf{r})/k_B T)$, where $k_B T$ is Boltzmann's constant times absolute temperature. Now let $C_q(\mathbf{r}) \geq 0$ be the average (bulk) concentration of either the $+q$ or the $-q$ ions at these positions; and $C_q(\mathbf{r}) = 0$ at positions

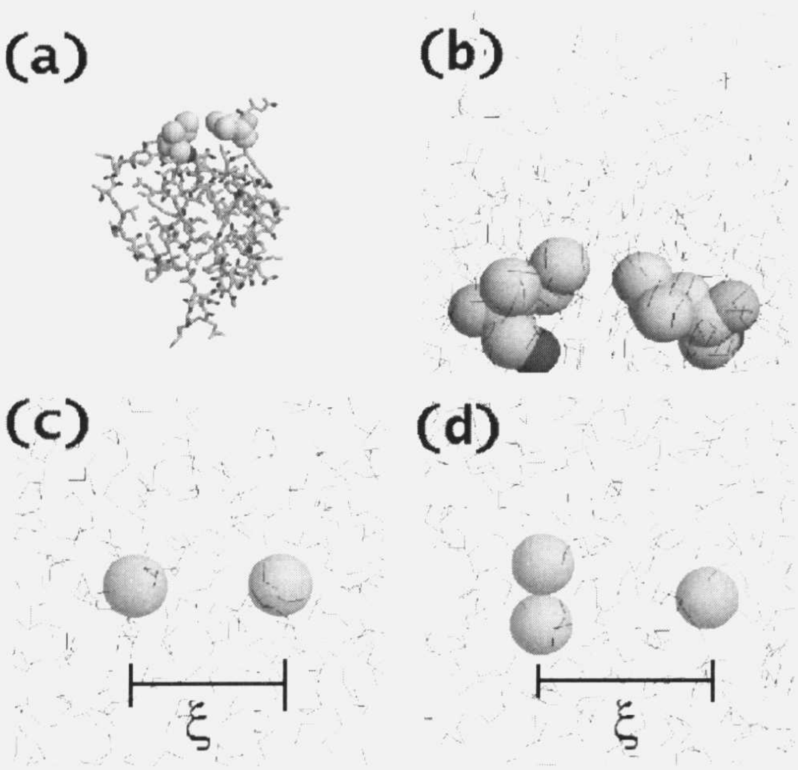


Figure 16.2
Schematics of hydrophobic interactions in protein energetics. (a) Native structure of chymotrypsin inhibitor 2 (protein databank entry 2CI2). As an illustration, two hydrophobic residues—a leucine at position 12 and an isoleucine at position 63—are highlighted by space-filling representations. (b) A blown-up view of the two hydrophobic residues in water. Water molecules are depicted by V-shaped representations, whereas the methyl and methylene groups of the residues are shown as spheres. (c, d) Modeling hydrophobic interactions among methyl and methylene groups by water-mediated interactions among methanes. Here each methane (sphere) is treated as a united atom, meaning that its nonpolar hydrogen atoms are not modeled explicitly. The distance ξ defines the two-methane (c) and three-methane (d) configurations for the potentials of mean force in figure 16.3.

inaccessible to mobile ions (in the core of a folded globular protein, for example). The bulk concentrations of the two oppositely charged ions are equal because the solvent is electrically neutral macroscopically. It follows that the mobile-ion contribution to the charge density is equal to $qC_q(\mathbf{r})[\exp(-q\phi/k_BT) - \exp(+q\phi/k_BT)]$.

The approximation introduced into this line of argument is not difficult to discern: The discrete ionic charge density (upon which an earlier step in the argument is premised) has been smeared out in the last expression to a continuous smooth distribution in the solvent. Consequently, possible sharp variations in $\phi(\mathbf{r})$ that might have resulted from discrete charges are precluded. The Poisson-Boltzmann equation is obtained by incorporating this approximate mobile-ion contribution into the Poisson

equation, viz.,

$$\nabla \cdot [\epsilon(\mathbf{r})\nabla\phi(\mathbf{r})] - \frac{2qC_q(\mathbf{r})}{\epsilon_0} \sinh\left[\frac{q\phi(\mathbf{r})}{k_B T}\right] + \frac{\rho_{\text{fix}}(\mathbf{r})}{\epsilon_0} = 0 \tag{16.2}$$

where the value of $C_q(\mathrm{r})$ in the solvent is proportional to its ionic strength (see discussions in Honig and Nicholls 1995; Gilson 1995; Dill and Stigter 1995; Roux and Simonson 1999; and references therein); and the (simpler) direct interaction case in the absence of mobile ions (figure 16.1b) corresponds to $C_q(\mathbf{r}) = 0$ for all $\mathbf{r}$. Equation 16.2 implies that $\phi(\mathbf{r})$ can be solved for any system with a given set of fixed charges, a solvent ionic strength, and a position-dependent dielectric constant. This continuum treatment of solvent charge effects has been applied to complex-shaped biomolecules to provide useful physical insight into, and graphical visualization of the role of electrostatic interactions in a wide range of biological phenomena (e.g., by using the DelPhi and GRASP programs; see Honig and Nicholls 1995).

16.3.2 Example: Hydrophobic Interactions

Solvent-mediated interactions can be modeled directly by empirical force-field simulations with explicit water and other solvent molecules. Explicit-solvent simulations take into account the shapes of discrete solvent molecules, giving predictions that are often more physical than that from continuum solvent models (Daura et al. 1998). Figure 16.3 shows recent explicit-solvent simulation results that bear on our understanding of hydrophobic effects.

Plotted in this figure are potentials of mean force[2] (PMF) among methane molecules in water (for recent reviews, see, e.g., Sorenson et al. 1999; Hummer et al. 2000). These systems have been investigated extensively as models for hydrophobic interactions in proteins because methane is chemically similar to the nonpolar methyl and methylene groups in hydrophobic amino acid residues (figure 16.2). Each PMF in figure 16.3 is a free energy function obtained from averaging over water configurations. It represents the sum total of solvent effects plus the direct interaction between the methanes, and it determines the relative probabilities of different methane configurations. Methane-water and water-water interactions lead to PMFs with spatial dependences that are more structured than that of the direct two-methane interaction (figure 16.3a). The salient features of two-methane PMFs are: (1) a contact minimum at $\xi \approx 3.8$ Å that is significantly more favorable to methane-methane association than that contributed

2. The *physical* potential of mean force considered here is different from the knowledge-based "potentials of mean force" (Sippl 1995) derived from database analyses.

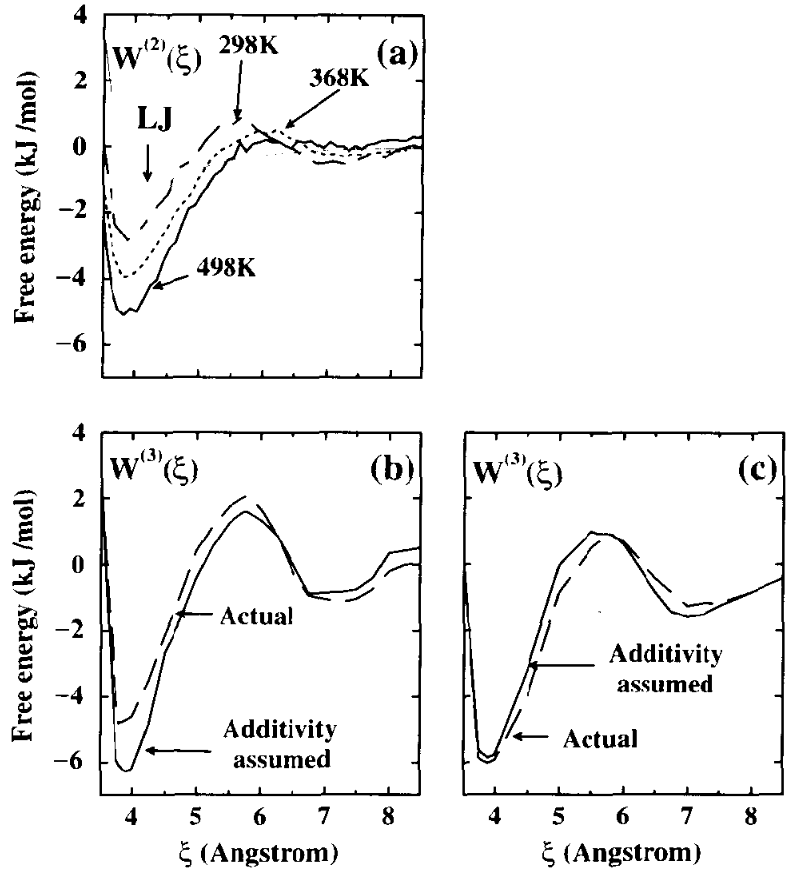


Figure 16.3
Hydrophobic potentials of mean force (PMF). The variable ξ in (a) is defined in figure 16.2c, whereas ξ in (b) and (c) is defined in figure 16.2d. (a) Two-methane PMFs, $W^{(2)}(\xi)$, simulated under atmospheric pressure at 298K (25 °C) and 368K (95 °C). Included for comparison are the direct Lennard-Jones (LJ) model potential between the two methanes, and a $W^{(2)}(\xi)$ simulated at an average water density of 829 kg/m^3 at 498K (225 °C) under constant volume conditions. Computational details are given in Shimizu and Chan 2000. (b) Three-methane PMF results from Rank and Baker (1997), adapted from their figure 4a. The free energy $W^{(3)}(\xi)$ plotted here is for bringing a methane from infinity to position ξ relative to the other two methanes that are already in contact. The "Actual" curve is the simulated three-methane PMF, $W^{(3)}(\xi)$. For each three-methane configuration specified by ξ, the sum of two-methane PMF values for the two pairs of methanes (between one of the contacting methanes and the third methane) is plotted as the "Additivity assumed" curve. (c) Corresponding three-methane PMF results from Czaplewski et al. 2000 for the same three-methane system as in (b), adapted from the 12-window plots of their figure 8d.

by the attractive direct methane-methane interaction alone, underscoring the role of water in promoting hydrophobic association; (2) a free energy barrier (desolvation peak) at $\xi \approx 5.7\,\text{Å}$; and (3) a second, shallow (solvent-separated) minimum near $\xi \approx 7.0\,\text{Å}$. Implications of these features on protein folding have recently been discussed (Shimizu and Chan 2000, 2001a; and references therein).

In protein folding energetics, a crucial question is to what extent interactions can be treated as additive (Mark and van Gunsteren 1994; Dill 1997; and references therein). This issue is pertinent to the question as to whether an appropriate scoring function for protein structure prediction can be devised using only pairwise additive amino-acid-based contact energies (see, e.g., Vendruscolo et al. 2000). Whether hydrophobic and other interactions are additive is also relevant to understanding protein calorimetric data (Chan 2000). Potential of mean force simulations can help address these issues. Here, figure 16.3b,c demonstrates that even if the underlying non-bonded atomic interactions of a model are *assumed* to be pairwise additive—as is the case for the force field used in these simulations—the resulting PMFs, which are *effective* interactions, are not necessarily additive. From the results of Rank and Baker (1997) in figure 16.3b, effective interactions among methanes in water appear to be anti-cooperative (Shimizu and Chan 2000c), at least for the interaction at the contact minimum. Free energy for bringing three methanes from infinity to contact is predicted by their simulation to be less favorable (less negative) than the sum of bringing three pairs of methanes together separately. On the other hand, the results in figure 16.3c from a different simulation of the same system by Czaplewski et al. (2000) implies the opposite—that effective contact interactions among methanes in water are slightly cooperative; in other words, the free energy for bringing three methanes from infinity to contact is more favorable (more negative) than the sum of bringing three pairs of methanes together separately. More recently, the discrepancy between these two studies has apparently been resolved. Using the same water model as that in Rank and Baker (1997) and Czaplewski et al. (2000) but a more reliable technique of analysis, simulation data from our group indicates that multiple methane hydrophobic interactions are largely anti-cooperative under ambient conditions. See Shimizu and Chan (2001b) for details.

Explicit-water simulations show that PMFs can be temperature dependent even when the underlying non-bonded atomic potential function of the model is temperature independent (figure 16.3a). Take two methanes in water as an example. The effective interaction $W^{(2)}(\xi)$ at each separation ξ between the two methanes involves an average over a huge number of different configurations of water molecules. The relative populations of these configurations are temperature dependent because not all configurations have the same energy. Hence $W^{(2)}(\xi)$ depends on temperature. This

fact is equivalent to characterizing the effective interaction as having an *entropic* part, whereas temperature-independent interaction may be referred to as being "purely enthalpic."

Temperature dependences of potentials of mean force have ramifications for recent all-atom simulations of protein unfolding kinetics. In an attempt to circumvent current computational limitations on all-atom simulations of protein folding (see above), all-atom *unfolding* molecular dynamics have been performed (Daggett 2000) at a high simulation temperature of 498K (225 °C) to speed up the unfolding process, using a significantly reduced average water density of 829 kg/m^3 (Daggett and Levitt 1992) compared to densities of 958–1000 kg/m^3 for water under atmospheric pressure between 0° and 100 °C. It has been asserted that kinetic protein folding pathways are temperature independent (Daggett and Levitt 1994), but this assertion is not valid in general. PMFs in aqueous solutions are sensitive to temperature and average water density (Shimizu and Chan 2000). Just as the PMF at each position involves averaging many solvent configurations, the probability for a protein molecule taking any *microscopic* folding or unfolding pathway (as defined by the trajectories of all atoms in the protein but *not* the trajectories of solvent degrees of freedom) is a function of the probabilities of many possible trajectories of the solvent molecules consistent with the given kinetic pathway of the protein. Because the relative probabilities of solvent trajectories can shift as temperature is varied (because they can involve different energy barriers), the relative favorabilities of different protein folding and unfolding pathways can change with temperature. Figure 16.3a shows a two-methane PMF simulated under conditions that have been used for high-temperature unfolding molecular dynamics simulation. It has features significantly different from PMFs at ambient temperatures and pressure. The high temperature and low water density lead to a much more favorable contact minimum, and the desolvation peak and solvent-separated minimum are all but abrogated. These observations suggest that, though insight can be gained from high temperature unfolding simulations (Lazaridis and Karplus 1997; Alonso and Daggett 2000), caution should be exercised in their interpretation.

16.4 Simple Self-Contained Polymer Models of Proteins

Proteins are chain molecules. Chain connectivity, stiffness, and excluded volume impose significant constraints on protein behavior (Chan and Dill 1991). In addressing the physical forces in protein folding, self-contained polymer models are indispensable. By self-contained polymer models, we refer to theoretical constructs in which

the conformational distribution of a chain molecule is determined solely by the energetic components that are being considered explicitly. This is the most intuitive and straightforward approach to physical modeling. Our only reason for emphasizing it here is because the traditional discourse of protein energetics often involves non-self-contained constructs. Typically, denatured states in non-self-contained constructs are simply assumed to have certain a priori conformational distributions (partially unfolded, random-coil-like) that are not derived from the elementary interactions under consideration. As a result, logical connections between elementary interactions and predicted properties cannot be unequivocally established using non-self-contained constructs. (For details, see discussions in Chan 2000; Kaya and Chan 2000a.)

Obviously, all-atom protein models based on empirical force field such as equation 16.1 are self-contained polymer models. The ability to use these models to simulate biochemical processes is expected to be steadily enhanced by advances in computer technologies, improvements in simulation techniques (Duan and Kollman 1998), and the development of efficient conformational sampling algorithms (Hansmann and Okamoto 1999; Feldman and Hogue 2000). Nevertheless, brute-force all-atom simulation of folding is still beyond our reach (Daggett 2000). More fundamentally, it is not always straightforward to clarify the essential physics of a complex system from a huge amount of detailed simulation data generated by a large number of parameters. Therefore, to address general principles in protein folding, it is necessary to also construct and analyze self-contained polymer models with simplifed representations of protein chain geometries and intrachain interactions, as we have argued above. Research in the past decade has demonstrated that simplified models can lead to novel concepts and provide useful insight (Bryngelson et al. 1995; Dill et al. 1995; Hinds and Levitt 1996; Thirumalai and Woodson 1996; Pande et al. 1997; Shakhnovich 1997; Chan and Dill 1998). But they also raise new questions, especially with regard to their relations to real proteins.

16.4.1 Lattice Representations of Chain Geometries

Simple protein models use reduced representations of the polypeptide chain. Chains in most of the recent simple models are configured on regular lattices. Simple off-lattice protein models with chains configured in the continuum (see e.g., Sun 1993; Sun et al. 1995; Hao and Scheraga 1998; Thirumalai and Klimov 1999; Klimov and Thirumalai 2000; Irbäck et al. 2000; and references therein) and off-lattice discretized conformational spaces (Park and Levitt 1995) have also been investigated. Owing to space limitations, only lattice models are discussed below.

The basic components of typical lattice chain models are shown in figure 16.4. Lattice models are more tractable because the number of possible chain conformations

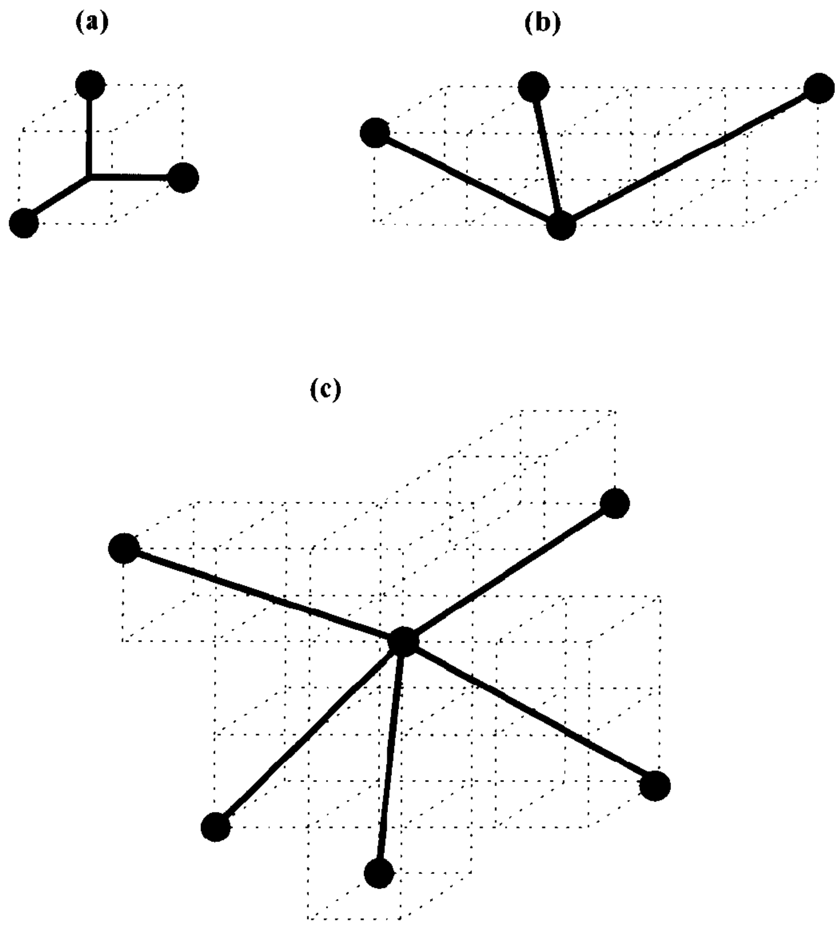


Figure 16.4
Lattice representations of protein chains. Example bond constructions are drawn as thick lines joining two solid circles. The dotted lines show the underlying mesh. (a) Simple cubic lattice models. (b) The hydrid-210 model. (c) The ultra-310 model. Bonds in the high-coordination lattice models (b) and (c) correspond to protein C_α-C_α virtual connections (Godzik et al. 1993).

is restricted by lattice regularities, allowing for faster searches and broad coverage of the discretized conformational space. Many recent lattice protein chain models are configured on three-dimensional simple cubic lattices (figure 16.4a). A further simplification is to configure them on two-dimensional square lattices. Both of these extremely simplified representations are introduced to capture chain connectivity, excluded volume, and certain general features of intrachain interactions (see below). They do not attempt to model geometric details of polypeptide chains. In most applications, the nodes along these lattice chains are not specified to correspond to any particular position of a real polypeptide; hence, they are not used as models for specific proteins, but rather as theoretical tools to address general properties of generic pro-

teins (Abkevich et al. 1994; Chan 1998a).[3] The rationale for this modeling approach is discussed in Dill et al. 1995.

To model protein structures at higher resolutions, extensive investigations have been undertaken by Kolinski, Skolnick, and coworkers to design lattice mimics of polypeptide geometries (Godzik et al. 1993; Kolinski et al. 1999). This effort began more than a decade ago with protein chain models on diamond lattices (Kolinski et al. 1987). Two of their models are shown in figure 16.4b,c, both of which use an underlying simple cubic lattice mesh to construct model polypeptide C_α–C_α virtual bond vectors. The unit length of the Cartesian coordinate system (i.e., the magnitude of vectors $[\pm 1, 0, 0]$ and their permutations) is equated with a length scale a to match model dimensions to real proteins. Different bond-vector construction schemes require different values of a. In an early "knight's walk" model, C_α–C_α virtual bonds were constructed from the 24 permutations of the vectors $(\pm 2, \pm 1, 0)$, and a was set to 1.70 Å so that the model virtual bond length $\sqrt{5}a = 3.80$ Å matches that of the predominant *trans* peptide bond; and amino acid side chains were treated as on-lattice entities (Skolnick and Kolinski 1991).

In a subsequent "hybrid"-210 system (figure 16.4b), C_α–C_α virtual bonds are constructed from the 56 permutations of the vectors in the set $\{(\pm 2, \pm 1, 0), (\pm 2, \pm 1, \pm 1), (\pm 1, \pm 1, \pm 1)\}$, and $a = 1.70$ Å (Kolinski et al. 1993). This was followed by a finer (less coarse-grained) "ultra"-310 system (figure 16.4c) that constructs C_α–C_α virtual bonds from the 90 permutations of the vectors in the set $\{(\pm 3, \pm 1, 0), (\pm 3, \pm 1, \pm 1), (\pm 3, 0, 0), (\pm 2, \pm 2, \pm 1), (\pm 2, \pm 2, 0)\}$, with $a = 1.22$ Å (Godzik et al. 1993; Kolinski and Skolnick 1994). In more recent applications, amino acid side chains were incorporated as off-lattice entities (Kolinski and Skolnick 1994). To conform to polypeptide geometries, restrictions are placed on the directions of consecutive virtual bonds to exclude some acute and open bond angles. Virtual bond angles are confined to the range of 78.5–143.1° and 72.5–154° for the hybrid-210 and ultra-310 models, respectively (Kolinski and Skolnick 1994). These "hybrid" and "ultra" systems are "fluctuating bond" models, in that C_α–C_α virtual bond lengths are not fixed within a model. In units of a, possible virtual bond lengths are $\sqrt{3}$, $\sqrt{5}$, $\sqrt{6}$ for the hybrid-210 model, and $\sqrt{8}$, 3, $\sqrt{10}$, $\sqrt{11}$ for the ultra-310 model. The values of a were chosen so that the predominant virtual bond length of 3.80 Å for real proteins corresponds roughly to the average virtual bond lengths in these models. Bond fluctuation models have been used to model polypeptides in other contexts (see, e.g., Chen 2001). Of particular interest is a recent fluctuating-bond "side-chain-only" lattice protein model,

3. It is interesting to note that real non-protein chain molecules have recently been designed for lattice-like folding in two dimensions (Choi et al., 2000).

in which 646 possible model virtual bonds with lengths ranging from 3.8 Å to approximately 10 Å are used to connect side-chain centers of mass instead of C_α's (Kolinski et al. 1999).

High-coordination lattices are quite flexible. Because many bond angles are allowed, conformations constructed on these lattices can match quite closely any polypeptide conformation at the C_α level. Average root-mean-square deviation of fitted lattice conformations from C_α traces of Protein Databank structures is ~1.0 Å for the hybrid-210 model, and ~0.8 Å for the ultra-310 model and the recent side-chain-only model (Godzik et al. 1993; Kolinski and Skolnick 1994; Kolinski et al. 1999).

16.4.2 Interaction Schemes in High-Coordination Lattice Models

Interaction schemes for high-coordination lattice protein models use knowledge-based parameters statistically derived from protein native structure databases as well as postulated potential functions. These schemes have evolved over the years (see, e.g., Skolnick and Kolinski 1991; Kolinski et al. 1993; Sikorski et al. 2000). Typically, their potential functions include energetic contributions from: (1) local orientation correlations between amino acid side chain rotamers; (2) lattice-defined hydrogen bonding interactions; (3) amino-acid-specific one-body terms (here a "body" refers to an amino acid residue) that depend on a residue's distance from the center of mass of a given protein conformation, or the number of contact a residue made with other residues; (4) pairwise (two-body) terms; and (5) four-body "tertiary interaction" terms to promote certain preferred side chain packing patterns. Parameters for the assumed functional forms of (1)–(4) were derived from protein databases, whereas (5) was postulated "by hand." Other additional features have also been incorporated (Sikorski et al. 2000; Kolinski et al. 1993).

Even with their rather complex chain representation and interaction schemes, these lattice models are computationally more tractable than all-atom models, and have provided insight into folding thermodynamics and kinetics (Kolinski et al. 1999; Sikorski et al. 2000), many of which cannot be addressed by other current models at comparably high levels of structural resolution. Physical interpretation of high-coordination lattice model results, however, is not always straightforward. This is because of the large number of tunable knowledge-based and postulated parameters involved, whose relationships with physical forces are sometimes not entirely clear. For instance, a "one-body" term described above postulates a gravitation-like pull on some residues toward a certain attractive center in a protein. But physically the collapse of a protein chain must originate from the solvent-mediated interactions among the residues, and therefore should be describable by atomic interactions, or two- and higher-body interactions at the residue level. It is not certain whether a

universal high-coordination lattice interaction scheme that can describe a broad range of protein properties would ultimately emerge.

16.4.3 Why Study Simple Low-Coordination Lattice Protein Models?

A complementary route to physical understanding is to adopt an incremental approach (Rapaport 1997; Chan 2000) that, as a first step, seeks to establish a workable conceptual framework to account for general properties of proteins. For this purpose, highly coarse-grained chain representations, such as those on simple cubic and square lattices, are used to capture only the rudimentary chain nature of protein but not their structural details. The *postulated* interactions in these models are simple, but can nonetheless be based on physical considerations, in a spirit very much akin to that of Ising models for ferromagnetism and related phenomena (Pathria 1980). These models have an important logical advantage because they are readily falsifiable. Their results are essentially direct deductions from the basic axioms of a given model's energetics, derived without intervening approximations and additional assumptions. (Unlike more complex models in which ad hoc approximations are often introduced for computational expediency.) Most of these models consider only contact interactions (see below), but orientation-dependent interactions can also be incorporated (Borg et al. 2001). Because of their highly simplified nature, extra care is needed to assess whether the basic interaction scheme of a given model warrants certain conclusions about real proteins, especially with regard to structural specifics and microscopic mechanisms of protein folding. Several such questions are raised below. Nonetheless, provided that both the advantages and shortcomings of simplified modeling are taken seriously, much can be learned about protein energetics from these exercises. This is underscored by the fact that although there can be many designs for highly simplified chain models, obtaining predictions consistent with experimental observations is nontrivial (Chan 1998b, 2000; Chan and Dill 1998; Kaya and Chan 2000a, 2000b). With proper applications of experimental constraints, simple protein model construction is not as arbitrary as it might seem.

16.4.4 Interaction Potentials in Simple Lattice Models: What Do They Represent?

Gō and HP+ Models The "Gō potential" is one of the first interaction schemes used in simple lattice models (Taketomi et al. 1975). Starting with a target native structure, a Gō model assigns equal favorable energies (< 0) to all contacts between a pair of residues (monomers) that occur in the given target structure, and assigns neutral energies ($= 0$) to all non-native contacts that do not belong to the target structure. Recent variants include an "HP+" model that assigns repulsive (> 0) instead of neutral energies to nonnative contacts (Chan and Dill 1998). From a physi-

cal perspective, such *teleological* constructs (Kaya and Chan 2000a) may appear trivial because they do not seek to account for the energetic favorability of the native state in terms of universal elementary physical interactions but, instead, fabricate a different interaction scheme for each different target structure. Because of their non-universality, Gō and Gō-like potentials do not provide a model solution to the most basic question in protein folding, which is how the amino acid sequence of a protein determines its native structure.

Having said that, in view of our severely limited knowledge at present, it is profitable to test all kinds of assumptions and ask "*what if*" questions whenever possible. Such exercises are useful because they allow us to partially sort out the many logical relationships that may lead to further progress, even though the physical origin of the assumptions remains to be elucidated. Pursuing nontrivial implications of Gō-like interactions can be fruitful within such a conceptual framework. In fact, currently some generic protein properties can only be qualitatively reproduced by such highly artificial constructs (Chan 1998b; Chan and Dill 1998); and results from some Gō model studies can provide remarkable and unexpected information (Micheletti et al. 1999; Clementi et al. 2000). Gō-like native-centric approaches have also been applied to model protein folding in non-lattice contexts (Alm and Baker 1999b; Galzitskaya and Finkelstein 1999; Muñoz and Eaton 1999; Zhou and Karplus 1999). However, it should always be borne in mind that these approaches by themselves do not tell us what physical interactions can conspire to create the highly specific "molecular recognition" features they assume. Answers to such basic questions have to be sought in studies that attempt to model the physical driving forces in proteins.

HP Models A widely applied simple lattice protein potential designed to capture physical driving forces is that of the HP model. It uses an extremely coarse-grained two-letter folding alphabet: sequences are strings of monomers that are either H (hydrophobic) or P (polar). (These model monomer names should *not* be confused with the one-letter codes for histidine and proline!) To mimic hydrophobic effects, each nearest-neighbor contact between two H monomers not consecutive along the sequence is assigned a favorable energy, irrespective of whether the contact is in the native (ground-state) structure or not; all other contacts are modeled as neutral (Lau and Dill 1989; Dill et al. 1995). Thus, ground-state structures of HP sequences are determined by a universal model potential. This contrasts with the Gō-model approach, which uses a particular potential for each target native structure. Example HP sequence and ground-state structures are given in figure 16.5a,d. Other two- and three-letter potentials have also been investigated (see discussion in Kaya and Chan 2000a).

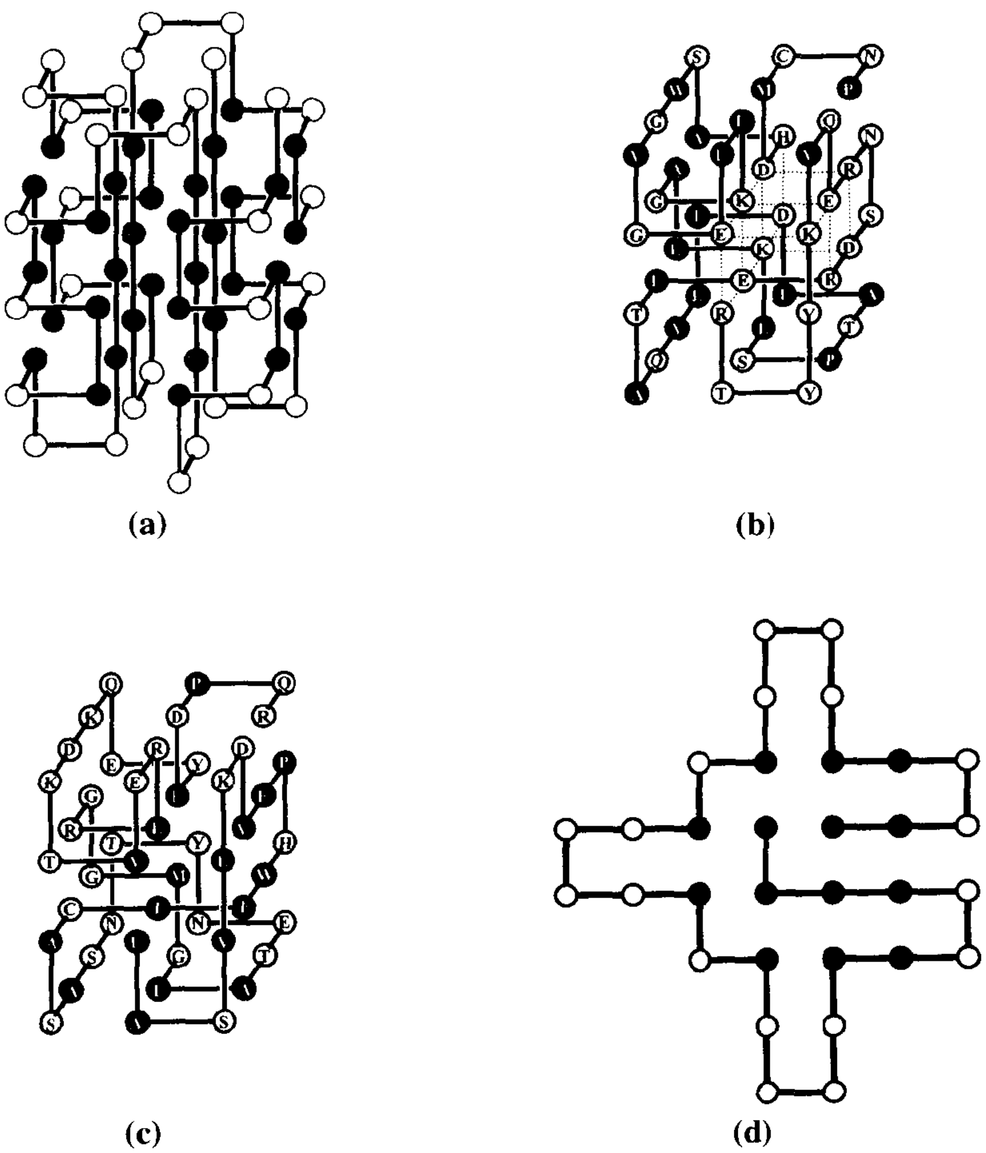


Figure 16.5
Simple lattice protein model examples. (a) One of three ground-state (lowest-energy) conformations of a 67-monomer (67mer) three-dimensional HP model sequence (Yue and Dill 1995). H and P monomers are depicted as filled and open circles, respectively. (b) The conformation with the lowest simulated energy for the 20-letter 48mer sequence shown (as given by the one-letter codes for the amino acids), computed with a particular set of interaction parameters modified by Shakhnovich et al. (1996) from table VI of Miyazawa and Jernigan 1985. Black circles are used for monomers corresponding to eight amino acid types (A, V, L, I, M, P, F, W) that are customarily considered to be hydrophobic. White circles are used for monomers corresponding to the other 12 amino acid types, including D, E, K, R, and H that are customarily referred to as charged. Each dotted line indicates a salt-bridge-like contact between a pair of monomers with opposite charges. (c) Same conformation as in (b), but for a different 20-letter 48mer sequence governed by a different set of interaction parameters. This conformation has the lowest simulated energy for the given sequence according to a set of contact energies modified (Shakhnovich et al. 1996) from the two-body interaction parameters of Kolinski et al. (1993). (d) A 32mer two-dimensional HP model sequence in the conformation with the lowest simulated energy (Irbäck et al. 1998).

The HP model potential is quite nonspecific and interaction heterogeneity is not high (Wolynes 1997). As a result, ground-state degeneracy (number of conformations with the same lowest energy) is generally high for HP sequences. The fraction of short HP sequences with up to approximately 20 monomers that have a unique ground-state conformation is ≈ 2.5 percent on two-dimensional square lattices (Chan and Dill 1996a). Degeneracy is higher in three dimensions. Determining the ground-state conformations of long HP sequences is a computational challenge (Yue et al. 1995) because in general it is an NP-complete problem (Berger and Leighton 1998; Crescenzi et al. 1998). No HP sequence configured on simple cubic lattices has been found to possess a unique ground-state conformation (Yue and Dill 1995; Yue et al. 1995). Recently, Buchler and Goldstein (2000) suggested that certain structural conclusions from HP model studies (see, e.g., Li et al. 1996) may not be general because these features are sensitive to the size of the alphabet and the particular energy model used. (For further discussion, see Tang 2000.) Limitations of HP-like 2-letter models have also been identified in theoretical investigations of coarse-graining techniques for reducing larger folding alphabets to smaller ones (Backofen et al. 1999; Micheletti et al. 1998; Wang and Wang 1999, 2000). These observations raise a fundamental modeling question: To what extent are HP and other simple lattice potentials proteinlike?

Despite its obvious limitations, the HP potential is useful in the study of proteins, for the following reasons.

First, in light of a host of simplifications in any simple model, a model folding alphabet is not necessarily less proteinlike solely because it has two instead of 20 letters. Other physical issues beside alphabet size can be equally, if not more, critical. These include the nature of a model's packing forces, namely the model interactions' specificity and stabilizing mechanisms (see below), the discriminating role of repulsive interactions, and whether the model conformational ensemble is unphysically restricted to only the maximally compact conformations (see dicussions in Chan and Dill 1996a, 1998; Backofen et al. 1999; Chan 2000).

Second, HP models should be useful for exploring the mapping between protein sequences and their native structures inasmuch as different energetic components contributing to protein folding (hydrophobic and other interactions) have no significant conflict with one another in the native conformation, that is, inasmuch as the situation envisioned by the consistency principle (Gō 1983) or principle of minimal frustration (Bryngelson and Wolynes 1987) is valid. In that case, we may adopt the working assumption that for a sequence to be proteinlike, it must have only a unique most-favored conformation or a very small set of near-unique most-favored conformations based on its hydrophobic-polar pattern alone, notwithstanding the fact that contributions from other energetic components have to be incorporated to fully

account for its thermodynamic and other properties (Kaya and Chan 2000a, b). This formulation has been motivated by the observation that symmetries exhibited by many HP model ground-state structures are intuitively more proteinlike (Yue and Dill 1995; Li et al. 1996) than the ground-state conformations favored by other model potentials, including some with larger alphabets (c.f. figure 16.5).

In this view, the unique model ground-state conformations determined by an HP (Lau and Dill 1989) or HP-like (Li et al. 1996; Hirst 1999) potential are adopted as coarse-grained models of protein native conformations, but with the understanding that additional *consistent* stabilizing driving forces may have to be invoked in applications that address more refined features of protein energetics (Chan 2000; Kaya and Chan 2000b). The viability of the present physical interpretation is buttressed by the recent finding that hydrophobicity patterns in two-dimensional HP model proteins as characterized by mean-square block fluctuation (Irbäck et al. 1996) are indeed qualitatively similar to that found in real protein sequences (Irbäck and Sandelin 2000).

Twenty-Letter Models Simple lattice models that use 20-letter alphabets have also been extensively studied (Shakhnovich 1997). As for the HP model, ground-state conformations in these models are determined by a universal set of interaction parameters (figure 16.5b,c). Typically, the interaction parameters are modified from knowledge-based pairwise contact energies between amino acid residues (Tanaka and Scheraga 1976; Miyazawa and Jernigan 1985, 1996), which are often represented as a symmetric 20×20 energy matrix with 210 independent elements. In general, knowledge-based statistical potentials can be quite different from physical interactions (Thomas and Dill 1996; Mirny and Shakhnovich 1996), although under certain restrictive conditions the former statistically derived parameters can provide a reasonable description of the underlying physics (Zhang and Skolnick 1998). Figure 16.6 provides one such example.

Here we analyze the 1985 version of the Miyazawa-Jernigan (MJ) contact energies (Miyazawa and Jernigan 1985, upper half of their table V) by a matrix diagonalization technique (Chan 1999). This method is a variation of an earlier analysis of Li et al. (1997). Shown on the left in figure 16.6a is the spectrum of eigenvalues of this particular MJ matrix. One of the eigenvalues is favorable (< 0) and dominant ($= -64.7$); others have much smaller magnitudes, between -1.93 and $+3.42$. In this formulation, the eigenvector of a given eigenvalue corresponds to a hypothetical linear combination of amino acid residues that has only one nonzero interaction, namely with itself; whereas its interactions with all other eigenvectors are zero. It follows that when a model system has a dominant eigenvalue, which may be inter-

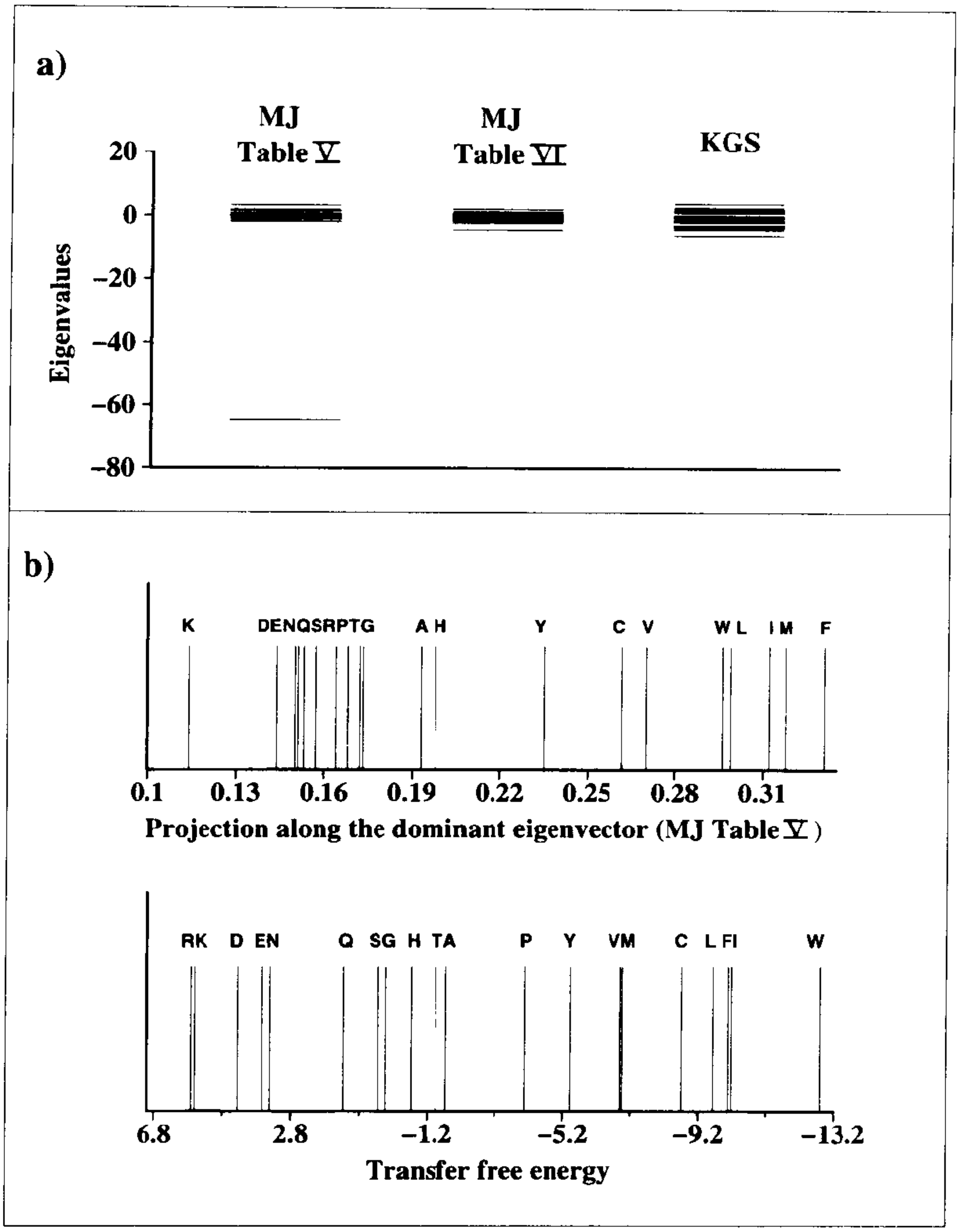


Figure 16.6
Matrix analyses of statistical contact energies."MJ table V" and "MJ table VI" here and in the text refers to the upper half of table V, and table VI of Miyazawa and Jernigan 1985, respectively. The "KGS" pairwise interaction parameters are from table III of Kolinski et al. 1993. The eigenvalues of these three matrices are given in (a). The upper plot in (b) shows the amino-acid components of the eigenvector of the dominant eigenvalue of MJ table V. See text and Chan 1999 for details. The lower plot in (b) is an hydrophobicity scale from Fauchère and Pliška 1983, in which an amino acid's experimental preference to be in octanol rather than in water is given by a transfer free energy in units of kJ/mol.

preted as a dominant mode of interaction, the interaction of the system may be approximated by the interaction involving only the eigenvector of the dominant eigenvalue. The projection of each amino acid along this dominant eigenvector (upper plot in figure 16.6b) may then be interpreted as the given amino acid's participation in the dominant mode of interaction.

Figure 16.6b strongly suggests that the main physical interactions captured by the MJ parameters are the "hydrophobic interactions," as has been pointed out previously (Godzik et al. 1995; Li et al. 1997). This is because the magnitudes of the amino acids' projections on the dominant eigenvector (upper plot in figure 16.6b) correlate quite well with amino acid hydrophobicity scales determined from water/oil solute transfer experiments (see, e.g., Karplus 1997; DeVido et al. 1998; Chan 2001), one of which is included here (lower plot in figure 16.6b). In general, residues that are more hydrophobic (i.e., those with lower water-to-oil transfer free energies) have larger components along the dominant eigenvector. These results are very similar to that obtained earlier (Chan 1999) for the 1996 version of this MJ matrix (Miyazawa and Jernigan 1996, upper half of their table 3), the latter is the physical basis for a recent novel algorithm for designing optimal sets of reduced amino acid alphabets with fewer than 20 letters (Wang and Wang 1999).

These findings are consistent with the original interpretation, based on a quasi-chemical approximation, of MJ table V as an approximate physical description of solvent-mediated interactions between amino acid pairs (Miyazawa and Jernigan 1985). In view of this, it is noteworthy that the MJ parameters in Shakhnovich and coworker's 20-letter model (Shakhnovich 1994) are not from MJ table V but from MJ table VI.[4] The latter was derived from MJ table V by shifting contact energies by a certain amount. Physically, this procedure may be viewed as the adoption of a new solvent or "reference state" different from that of the original system (Jernigan and Bahar 1996). In general, just like changing the solvent in a protein experiment can change the protein's conformational distribution, "shifting" a set of energy parameters can change the physics of a model system (Chan and Dill 1996a; Hirst 1999; Giugliarelli et al. 2000). This is clearly demonstrated by the fact that, unlike that for MJ table V, the spectrum of eigenvalues for MJ table VI lacks a dominant eigenvalue (figure 16.6a). Because the 20-letter model of Shakhnovich et al. is the foundation of a large body of interesting work (Shakhnovich 1997), we now take a closer look at the physical implications of its interaction parameters.

4. All 210 MJ interaction parameters they use are from MJ table VI, except three entries they chose to modify (see footnote on page 342 in Chan and Dill 1996a).

One conspicuous consequence of using MJ table VI is the prevalence of salt-bridge-like charge interactions in the cores of model native conformations, as in figure 16.5b. Instead of a proteinlike hydrophobic core, the core of this model sequence is made up entirely of charged monomers. All hydrophobic monomers are on the surface, none is inside. In short, compared to real proteins, the native structure of this model has an "inside-out" hydrophobicity distribution.

Does this matter? Whether this ostensively non-proteinlike feature is a cause for concern depends on what knowledge about proteins we are seeking from these models. If our interest is restricted to how highly coarse-grained protein properties may arise from general properties of heteropolymers, we may view the use of MJ table VI just as a convenient way of implementing a model interaction scheme with sufficient heterogeneity, without regard as to whether its details are proteinlike or not. The class of 20-letter models in question can be instrumental in such investigations. In that case, however, we should refrain from, or at least be very conservative about identifying detailed features of the model with structural or mechanistic aspects of real proteins. By contrast, if we are interested in how physical interactions in proteins affect folding mechanisms, it is only logical to inquire to what degree the model interaction scheme coincides with the driving forces in proteins. If this is one's goal, it is necessary to ascertain whether the very nature of the physical interactions has been changed by using MJ table VI instead of MJ table V.

To get to the root of this matter, we look beyond the nominal identities of the model monomers and focus on the mathematical properties, namely the signs and magnitudes, of the contact energies themselves. The native core of the 20-letter model in figure 16.5b is stablilized by a network of 15 contact interactions between nominally charged monomers (dotted lines), all but one of which are contacts between negatively charged monomer types D, K and positively charged monomer types K, R. There are 6 E-K, 3 D-K, 3 E-R, and 2 D-R contacts. According to MJ table VI, the energies of these contact types are -0.97, -0.76, -0.74, and -0.72, respectively.[5] Second only to the -1.06 value for a pair of cysteines, these are the next four most favorable interactions in MJ table VI. The (unweighted) average of these four strongly attractive contact types in the native state equals -0.80. On the other hand, the corresponding average for the six contact types among D, E, R, and K that *do not* appear in the native structure is $+0.16$, as they are either intermediately to strongly

5. The energies in the present discussion are read off directly from MJ table VI. In their analysis, Shakhnovich et al. (1996) use a set of modified parameters obtained by shifting MJ table VI so that the average of all resultant parameters equal zero. This amounts to adding -0.018 to every energy parameter in MJ table VI. Because the magnitude of -0.018 is small compared to almost all of the parameters, the conclusions reached here are independent of whether this shift is used or not, as can be readily verified.

repulsive or essentially neutral to slightly attractive: according to MJ table VI, the D-D, D-E, E-E, R-R, R-K, and K-K contact energies are +0.04, −0.15, −0.03, +0.11, +0.75, and +0.25, respectively. This implies that the interactions among these four monomer types in this model do have features similar to that of electrostatic interactions, although MJ table VI provides for stronger attractions between opposite charges than the repulsions between like charges.

This analysis reveals the mechanism by which alternate nonnative core packing arrangements are energetically disfavored in this model. Native stability is the outcome of a competition between native and nonnative configurations. Structural specificity and thermodynamic stability of the native structure increase if compact nonnative configurations are disfavored. The above observation means that this model's relatively high degree of interaction specificity originates from an ionic-crystal-like interaction pattern in its native core. Interactions with high specificities are likely to be needed to account for real protein behaviors (Chan 2000; Kaya and Chan 2000b). However, the currently accepted physical picture is that nonlocal electrostatic interactions, whose features the model in figure 16.5b appears to be capturing, do not play a significant role in stabilizing the protein core (Dill et al. 1990; Honig and Yang 1995).

The interaction patterns of some other 20-letter lattice models are more protein-like, at least ostensively. Examples include models based on KGS parameters (figure 16.5c). The two-body KGS interaction parameters of Kolinski et al. (1993) were not intended by the original authors to be used as the sole contribution to the energy of a model protein, because there are many other terms in their potential. The distribution of KGS eigenvalues is quite similar to that for MJ table VI but not the hydrophobicity-dominant MJ table V (figure 16.6a), presumably because hydrophobic effects are described mainly by the one-body term in the KGS formulation. Nonetheless, the core of the conformation in figure 16.5c is made up of hydrophobic monomers, with 21 contacts among A, V, L, I, M, F, and W, encompassing 10 of the 28 possible pairings among these monomers. The KGS two-body parameters stipulates that all these 28 contact types are favorable except one (the slightly unfavorable +0.1 for A and M).[6] The (unweighted) average energy of the 10 pairwise interaction types that occur in the native structure is −0.87, which is only slightly more favorable than the average of −0.68 for the 18 pairwise interaction types that do not appear in the native conformation. Hence the interaction pattern stabilizing

6. In Shakhnovich et al. 1996, the KGS parameters are shifted so that the resultant parameters average to zero. This amounts to adding a small number +0.15 to all the KGS energies used in the present discussion. Because this amount is small compared to most KGS energies, the effect of this shift on our conclusions is negligible.

the native core of this model may be characterized as low-specificity hydrophobic interactions, which are believed to be a significant contributing factor to protein native stability (Dill 1990; Honig and Yang 1995). A similar example is a 24mer model sequence of Betancourt and Thirumalai (1999b), whose native core is stabilized by eight contacts among V, L, and A monomers. Consistent with the usual notion of hydrophobicity, their interaction parameter set (Betancourt and Thirumalai 1999a) stipulates that contact energies between all possible pairs among V, L, and A are favorable. These include contact pairs that do and do not occur in the model native conformation in question.

These observations suggest that the stablization mechanisms of the latter two 20-letter models are quite different from that of Shakhnovich and coworkers. Besides figure 16.5b, other native structures in the model of Shakhnovich et al. also tend to have nominally charged monomers buried in their cores, though it is sometimes possible to achieve similar results with less stable model sequences that bury nominally hydrophobic monomers (Abkevich et al. 1995a). The above analysis shows that the issue at hand is not a trivial semantic quibble about the nominal identities of the model monomers, because in essence it is about the underlying mathematical and physical properties of the model potential function that determine folding mechanisms and a model's ability or inability to predict real protein properties. The 20-letter model in question has provided insight in many applications (Shakhnovich 1997), but apparently this model fails to capture some key features of protein energetics. An indication is that although the importance of native-structure topology to protein folding is generally recognized by its authors (Abkevich et al. 1994), this model has predicted a correlation trend (Abkevich et al. 1995b) between folding rate and relative contact order (which is a measure of the average sequence separation of contact residues) that is *opposite* to the observed experimental trend (Plaxco et al. 1998, 2000; see also Chan 1998a; Dinner and Karplus 1998). Significant advances have been made in the past decade in simple lattice protein modeling, but much work is still needed to achieve better matches between theory and experiment (Alm and Baker 1999a; Plaxco et al. 2000).

16.5 Recent Applications of Simple Lattice Protein Models

We now turn to more recent results. Because a comprehensive review is beyond the scope of this chapter, the topics covered below are necessarily selective and obviously biased by our own research interests. A major purpose in presenting them here is to illustrate how physical insight can be gained from simple modeling.

16.5.1 Mutations and Evolutionary Landscapes

Simple lattice protein models are useful for addressing conceptual issues in evolution. In general, a model of evolution requires a mapping from sequence (genotype) to fitness. Such a relation is often referred to picturesquely as a given model's evolutionary landscape. Biologically, a sequence's fitness is derived from its function, and function is intimately related to the structure(s) (phenotype) the given sequence encodes. Analytical models of sequence-fitness mapping (see, e.g., Perelson and Macken 1995) are instructive and useful in many respects. But these models do not address the underlying physical mechanisms by which sequences are mapped onto structures.

To tackle these questions, many theoreticians have adopted computational approaches, focusing on constructing models of sequence-structure mapping that are motivated by various aspects of polymer physics. To our knowledge, the first such effort was the seminal work of Fontana and Schuster (1987) on RNA secondary structures, which has led to an elucidation of continuous versus discontinuous evolutionary changes (Fontana and Schuster 1998). For globular proteins, mutations and evolutionary issues were addressed using the HP model (Lau and Dill 1989), which was applied early on by Lipman and Wilbur (1991) to investigate whether nonlethal mutations form a connected network. Since then, protein evolution has also been investigated using other simplifed chain models. These include approaches that employ sampling and/or enumeration of the full ensemble of chain conformations in on-lattice (Bussemaker et al. 1997; Mirny et al. 1998) and off-lattice (Nelson and Onuchic 1998) models, and analyses based on enumeration of maximally compact lattice conformations (see, e.g., Li et al. 1996; Govindarajan and Goldstein 1997), among others. (For an updated comprehensive review of analytical and computational approaches to protein evolution, see Voigt et al. 2001.)

Neutral Nets "Neutral net" is an evolutionary concept whose properties can be explored using these simple physical models. A neutral net is a collection of sequences interconnected via single-point mutations and encoding for the same ground-state structure (Bornberg-Bauer 1997; Bornberg-Bauer and Chan 1999; and references therein). An interesting idea emerging from these studies is that neutral net *topology*—the pattern of the sequences' interconnections in a neutral net—can have an effect on the distribution of steady-state evolutionary population among the sequences in the net. This hypothesis posits that, when all else is equal, sequences that have more connections to other sequences in the same neutral net are expected to be more populated in evolution simply by virtue of their mutational stabilities. As a determinant of evolutionary population, simple lattice model studies show that this neutral net topology

factor often acts in concert—as in the *superfunnel* scenario with a *prototype sequence*—but can also oppose the evolutionary force that arises from the relative functional fitness of individual sequences (Bornberg-Bauer and Chan 1999). Van Nimwegen et al. (1999) have independently arrived at a similar conclusion via a different modeling approach. In principle, this topology-population hypothesis should be testable by extensive mutagenesis experiments. Simple lattice model studies also show that sequences in a neutral net are often homologous (figure 16.7). Results from these investigations have been applied to provide theoretical underpinning (Cui and Wong 2000) for a recent technique that uses "averaged" information from a set of homologous sequences to reduce the statistical noise in knowledge-based potentials (Finkelstein 1998).

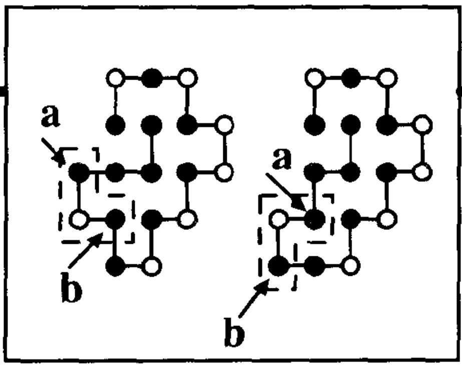
To illustrate the methods used in these studies, figure 16.7 lists the sequences of two HP model neutral nets obtained by exhaustive enumeration of all possible conformations (Bornberg-Bauer and Chan 1999). As in real proteins (Bowie et al. 1990), the HP model protein cores are made up of conserved hydrophobic monomers. The "bridge" sequence in the center box serves as an intermediate step (Bornberg-Bauer 1997) along a possible mutational path by which a sequence changes from encoding uniquely for one conformation to another (see caption for figure 16.7). Remarkably, some features of these HP model predictions are reminiscent of the elegant experiments recently performed by Sauer and coworkers (Cordes et al. 1999, 2000).

Cordes et al. found that interchanging a hydrophobic core residue and a surface polar residue in each of the two subunits of the Arc repressor homodimer changes a surface inter-subunit two-strand β-sheet in the wild-type native conformation to a pair of α-helices in the native conformation of what they called a "switch" mutant (Cordes et al. 1999). Analogously, in figure 16.7, a pair of mutations that interchanges the sequence positions of an H and a P results in a similar effect: starting with the unique sequence connected to the left side of the center box, the pair of substitutions P $\rightarrow$ H at site **a** and H $\rightarrow$ P at site **b** leads to a different local fold on the surface (dashed boxes) of the HP model protein, namely from the "L" shape on the left to the "Γ" shape on the right. More recently, Cordes et al. (2000) discovered that a sequence that is a single-substitution mutant of both the wild-type Arc repressor and the "switch" mutant have approximate equal populations in the wild-type native conformation and the "switch" mutant's native conformation. The role of this new Arc repressor mutant is thus analogous to that of the model "bridge" sequence in figure 16.7.

This set of experiments has important implications for evolution because it demonstrates the possibility that for some proteins a very small number of mutations can lead to a new fold. It shows that such processes can be determined in large measure

HPHPHPPHPHPPHPPHHH
HPPPHPPHPHPPHPPHHH
HPHPHPPHPHPPHPHHHH
HPHHHPPHPHPPHPPHHH
HPHPHPHHPHPPHPPHHH
HPHPHPPHPHPPHHPHHH
HPHPHHPHPHPPHPPHHH
HPHPHPPHPHHPHPPHHH
HPHPHPPHHHPPHPPHHH
HPHPHPPHPHPHHPPHHH
HPPPHPPHPHPHHPPHHH
HPPPHHPHPHPPHPPHHH
HPPPHPPHPHPPHHPHHH
HPHPHHPHPHPHHPPHHH
HPHPHPHHPHHPHPPHHH
HPPPHPPHPHPPHPHHHH
HPHHHPPHPHPHHPPHHH
HPHPHPHHPHPPHPHHHH
HPHPHPPHHHPPHPHHHH
HPPPHHPHPHPHHPPHHH
HPPPHPPHHHPPHPPHHH
HPPPHPPHPHHPHPPHHH
HPHPHHPHPHPPHHPHHH
HPHPHPHHPHPHHPPHHH
HPHPHPPHHHHPHPPHHH
HPHPHPPHPHPHHHPHHH
HPPPHHPHPHPPHHPHHH
HPHHHHPHPHPPHPPHHH
HPPPHPPHPHPHHHPHHH
HPHPHPPHPHHPHPHHHH
HPHHHPPHPHPPHHPHHH
HPHPHPPHPHHHHPPHHH
HPHPHHPHPHHPHPPHHH
HPHHHPPHPHPPHPHHHH
HPHHHPPHPHHPHPPHHH
HPHPHPPHPHPPHPPPHH
HPPPHHPHPHPHHHPHHH
HPHPHHPHPHPHHHPHHH
HPHPHPHHPHHPHPHHHH
HPHHHPPHHHPPHPPHHH
HPHHHPPHPHPHHHPHHH
HPPPHPPHPHHHHPPHHH
HPHPHPPHHHHPHPHHHH
HPHPHPHHPHHHHPPHHH
HPPPHHPHPHHPHPPHHH
HPPPHPPHHHPPHPHHHH
HPHHHHPHPHPHHPPHHH
HPPPHPPHHHHPHPPHHH

(a)



HPHPHPPHPHPHPPHHHH
HPPPHPPHPHPHPPHHHH
HPHHHPPHPHPHPPHHHH
HPHPHHPHPHPHPPHHHH
HPPPHHPHPHPHPPHHHH
HPPPHPPHPHPHPHHHHH
HPHHHHPHPHPHPPHHHH

(b)

Figure 16.7
A switch between two neutral nets. Sequences in an HP model neutral net are connected via a network of single-site H → P or P → H mutations (Bornberg-Bauer 1997; Bornberg-Bauer and Chan 1999). (a) The complete list of all two-dimensional HP sequences that constitute a neutral net in Bornberg-Bauer and Chan (1999; their figure 1a). Each of the 48 sequences encodes *uniquely* for the left conformation in the center box. (b) A smaller neutral net with seven sequences; each encodes *uniquely* for the right conformation in the center box. In both the (a) and (b) columns, sequences are listed from top to bottom in decreasing order of native stability. In each neutral net, the top sequence is the most stable; and conserved H or P monomers are shaded. Shown in the center box is a doubly degenerate sequence that encodes for both conformations and serves as a "bridge" between the two unique sequences connected by horizontal lines to the box. The bridge sequence differs from either of these two unique sequences in (a) and (b) by only a single-site substitution, at the specific sites **a** and **b** marked by arrows, respectively. Thus it may be viewed as an evolutionary link between the two neutral nets.

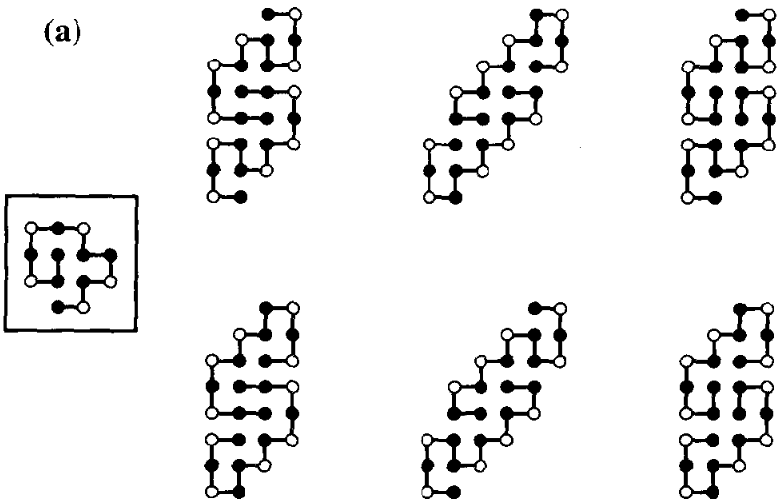
by the evolution of the hydrophobicity pattern of the protein sequence. In this regard, as we have argued above, even a minimalist notion of hydrophobicity as implemented in the HP model can be used to rationalize coarse-grained properties of the mapping between protein sequences and their native structures.

16.5.2 Protein Aggregation and Conformational Propagation

Because of their computational tractability and broad coverage of conformational space, simple lattice models can be used to investigate protein misfolding and aggregation. For instance, lattice models have been applied to explore the feasibility of a proposed iterative annealing mechanism of chaperonin action, which hypothesizes that a chaperonin functions by partially unfolding a misfolded protein—multiple times if necessary—so as to give the protein more chances to fold correctly (Todd et al. 1996; Lorimer 1997; Sigler et al. 1998). Lattice kinetics models of the proposed process have been used to study the physical factors involved (Chan and Dill 1996b; Sfatos et al. 1996; Betancourt and Thirumalai 1999b). More recently, simple lattice models have been applied to protein aggregation. In most cases, high-resolution modeling is not currently feasible for these multiple-chain processes. The thermodynamics and kinetics of two-chain systems have been studied using lattice models in both two (Harrison et al. 1999, 2001; Istrail et al. 1999) and three (Broglia et al. 1999; Harrison et al. 1999) dimensions (see figures 16.8, 16.9). Kinetics simulations of three-chain systems in two dimensions (Harrison et al. 2001) and a concentrated solution of up to 40 two-dimensional HP 20mer sequences (Gupta et al. 1998) have been conducted. Based on a presumed idealized packing geometry for an extensive two-dimensional aggregate state (figure 16.8b), Giugliarelli et al. (2000) investigated how aggregated chains may adopt a stable fold different from that of its single-chain native conformation.

Many of these studies have been motivated by "protein misfolding" diseases, including Alzheimer's, systematic amyloidoses, and prion diseases (Cohen and Prusiner 1998; Kelley 1998; Dobson 1999). Prion diseases of humans and other mammals are believed to be caused by a misfolded form PrP^{Sc} of the normal cellular prion protein PrP^{C} (Cohen and Prusiner 1998). Other prions in lower organisms have also been discovered (Lindquist 1997; Li and Lindquist 2000). Here we highlight several recent findings of Harrison et al. (1999, 2001). Their modeling identifies PrP^{C} with the ground-state (native) conformation of a single isolated model chain, whereas multiple-chain aggregates with two or more model chains in alternate folds different from that of the single-chain conformation are identified with PrP^{Sc}. Consistent with the proposed prion disease mechanism, they found that a non-negligible fraction of HP sequences adopt thermodynamically more stable alternate folds upon dimerization

(a)



(b)

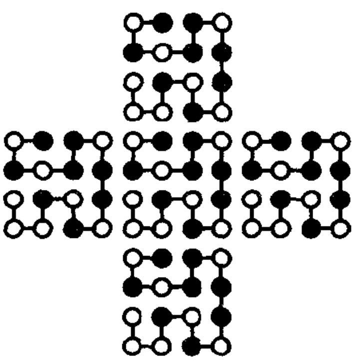


Figure 16.8
Models of protein aggregation. (a) In the box on the left is the unique ground-state (native) conformation with six HH contacts of a single isolated HP model chain. The same sequence gives rise to six different (two-chain) homodimers shown, each of which has 14 HH contacts. These energetically equally favorable homodimer configurations are obtained from extensive conformational enumerations (Harrison et al. 1999). They are thermodynamically more stable than any docking of two single-chain native conformations because the latter can at most have a total of 13 HH contacts. (b) A "super-lattice" of aggregated model proteins studied by Giugliarelli et al. (2000). This model is based on shifted forms of Li et al.'s (1996) modified HP potential.

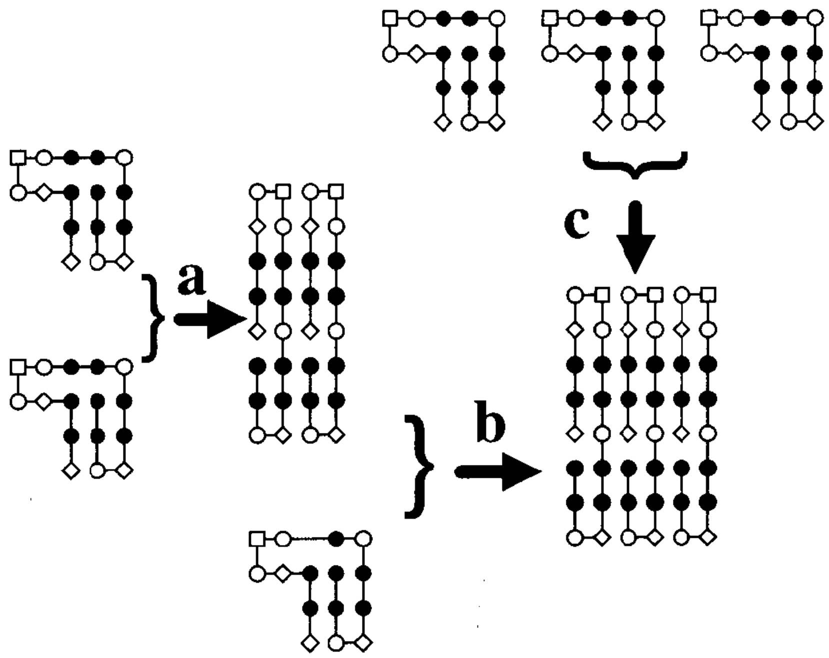


Figure 16.9
A simple 4-letter chain model of conformational propagation kinetics. Kinetic process **a** is a *spontaneous* two-chain conversion from two single-chain native conformations (on the left) to a more stable homodimer in which the chains adopt an alternate fold different from the single-chain native conformation. **b** is a *templated* three-chain conversion whereby a chain in the single-chain native conformation is placed near an already converted homodimer, then all three chains are allowed to refold in close proximity with one another to the more stable three-chain aggregate shown. **c** is a *spontaneous* three-chain conversion process whereby all three chains initially in the single-chain native conformation are converted to the same three-chain aggregate as in **b**. Simulated templated process **b** proceeds at approximately five times the rate of spontaneous process **c**; see Harrison et al. 2001 for details.

(figure 16.8a), and that model proteins whose single-chain native states are less stable are more susceptible to this form of misfolding. Multiple alternate folds that are equally stable can arise upon dimerization of the same sequence (figure 16.8a). This lattice scenario offers a perspective for understanding PrP^{Sc} strains, which have distinct self-propagating properties that are apparently encrypted in their different tertiary structures (Cohen and Prusiner 1998; Harrison et al. 1999).

A characteristic of the prion diseases is that apparently the disease-causing form PrP^{Sc} can serve as a template to lower the kinetic barriers that normally prevent PrP^{C} from converting to PrP^{Sc} (Cohen and Prusiner 1998). The lattice scenario in figure 16.9 suggests that multiple-chain kinetics with these postulated features are feasible (Harrison et al. 2001). This perspective suggests that, vis-à-vis the onset of prion diseases, the normal healthy condition corresponds to a situation in which the normal form of the prion protein is kinetically prevented from reaching thermodynamically

more stable disease-causing forms. Model simulation suggests that molecular crowding effects may be a main reason for the extremely slow kinetics. In other words, in normal situations, the conversion of PrP^C to PrP^{Sc} may have been arrested by kinetic traps, the physics of which is reminiscent of the sluggish dynamics in glassy materials (Kauzmann 1948; Bryngelson and Wolynes 1987; Bryngelson et al. 1995; Harrison et al. 2001).

16.5.3 Calorimetric Cooperativity as Modeling Constraint

One of the generic thermodynamic properties of small single-chain proteins is their calorimetric two-state cooperativity, that the ratio of their van't Hoff to calorimetric enthalpies $\Delta H_{vH}/\Delta H_{cal} \approx 1$ (Privalov 1979; Freire 1995; Makhatadze and Privalov 1995). Recently, it has been pointed out that this criterion, in conjunction with other experimental protein properties from small-angle X-ray scattering and NMR measurements, can be used to ascertain the extent to which a self-contained polymer model is proteinlike (Chan 2000; Kaya and Chan 2000a).

The main ideas are illustrated in figure 16.10, which shows that to satisfy the calorimetric two-state criterion ($\Delta H_{vH}/\Delta H_{cal} = \kappa_0 \approx 1$), the denatured density of states has to be very narrow (figure 16.10a). The heat capacity function C_P of a calorimetrically more cooperative model protein is sharper, whereas that of a less cooperative one has a long tail of appreciable non-zero contribution at high temperatures (figure 16.10b). The physical origin of the latter behavior is that the average enthalpy of the denatured population of a less cooperative or non-cooperative model protein undergoes a significant shift as temperature is raised (figure 16.10c), whereas the enthalpy distribution of a calorimetrically cooperative model's denatured population does not shift significantly over a wide range of temperature (figure 16.10d). Chan (2000) and Kaya and Chan (2000a, b) offer detailed analyses.

Nearly all popular contact energy based models, except three-dimensional Gō models, were found to fall short of meeting the calorimetric two-state criterion to various degrees (Chan 2000; Kaya and Chan 2000a, b). This suggests that some key energetic ingredients are missing in existing simple protein models. Therefore, the calorimetric two-state criterion should be applied as a constraint to facilitate the development of more proteinlike models. Preliminary analyses suggest that both local (Irbäck et al. 1997; Baldwin and Rose 1999) and nonlocal interactions (Dill 1990; Chan and Dill 1991; Yue and Dill 1996) and their cooperative interplay (Chan 2000; Kaya and Chan 2000b, and references therein) are necessary for calorimetric two-state cooperativity. This investigation has provided novel insight into theoretical and experimental aspects of native-state conformational diversity, including a clarification of the physical meaning (Kaya and Chan 2000a) of the multiple-conformation

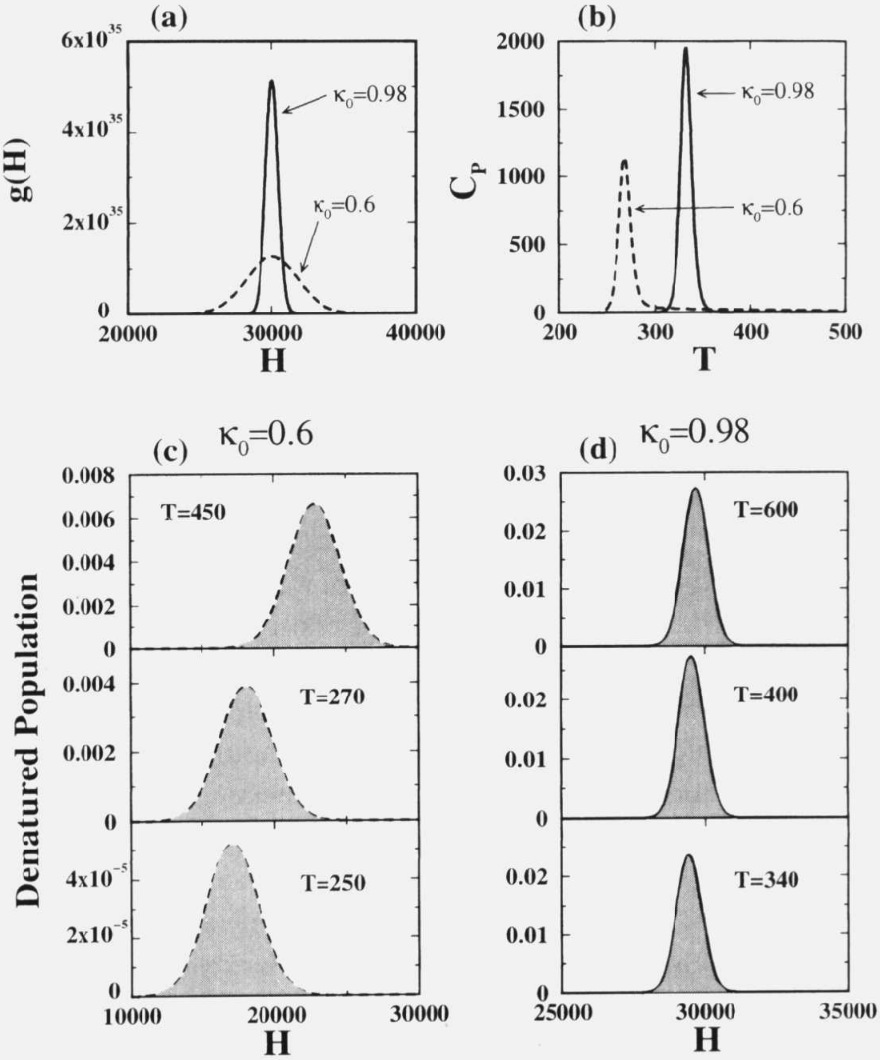


Figure 16.10
Density of states determines calorimetric cooperativity. Results are obtained using the random-energy Gaussian model in Chan 2000 and Kaya and Chan 2000a, which are based on experimental parameters from chymotrypsin inhibitor 2. Here $g(H)$, H, C_P, and T denote density of states, enthalpy in units of k_B, heat capacity in units of k_B, and absolute temperature, respectively; the native enthalpy is equal to zero; κ_0 is the population-based van't Hoff to calorimetric enthalpy ratio $\Delta H_{vH}/\Delta H_{cal}$ defined in Kaya and Chan 2000a. Here a model protein's density of states is its number of conformations as a function of enthalpy. Results for a high-cooperativity model ($\kappa_0 = 0.98$, solid curves) and a low-cooperativity model ($\kappa_0 = 0.6$, dashed curves) are compared. The curves in (a) are the denatured parts of the densities of states of the two models; and (b) shows their heat capacity functions. The two lower plots show the distribution of denatured-state enthalpy at the temperatures indicated, for the model with $\kappa_0 = 0.6$ (c) and $\kappa_0 = 0.98$ (d), respectively. The shaded areas under the curves are proportional to the total fractional denatured population. Note that different vertical scales are used for different temperatures. See Kaya and Chan 2000a for details.

native state defined in some 20-letter models (Gutin et al. 1998). Moreover, because the calorimetric criterion addresses energetic distributions among the unfolded non-native conformations, it also bears on the Z-score used in empirical protein structure prediction (see discussion in Chan 2000).

16.6 Issues in Simple Lattice Modeling of Folding Kinetics

Recent years witness extensive efforts in using simple lattice models to study protein folding kinetics. (For early efforts, see, e.g., Shakhnovich et al. 1991; Leopold et al. 1992; Miller et al. 1992.) We do not repeat their many findings here, as there is no shortage of comprehensive studies and reviews on the subject (see, e.g., Dill and Chan 1997; Onuchic et al. 1997; Shakhnovich 1997; Chan and Dill 1998; Thirumalai and Klimov 1999), including perspectives on a proposed nucleation folding mechanism (Shakhnovich 1998; Thirumalai and Klimov 1998). Here we only focus on modeling issues that we believe are basic, but that have received relatively little attention. For protein folding kinetics, in-depth analyses are often necessary to match simple lattice model results to experimental data. One noteworthy example is that temperature dependences have to be introduced into both the model intrachain interaction and the "Monte Carlo clock" of the model dynamics in order to provide a self-consistent account of the experimentally observed chevron plots and non-Arrhenius kinetics (Chan 1998b; Chan and Dill 1998).

Extra caution has to be used in kinetic than in thermodynamic applications of lattice models. The main reason is that, unlike continuum molecular dynamics, lattice dynamics are not governed by Newton's equation of motion. To mimic reality, model dynamics on lattices are designed with ingredients that are intuitively recognized to be physical features of real dynamics. It has long been realized that lattice protein folding kinetics models with local chain moves alone may have serious artifacts (Skolnick and Kolinski 1991). As a result, authors of high-coordination models have been careful in choosing moves and assigning physically plausible weights to the moves. To ensure that various possible mechanisms of protein assembly are not *a priori* excluded, they have designed their lattice moves in such a way that the model dynamics allow for the slow diffusion of assembled fragments of secondary and supersecondary structure, but at the same time the assembled fragments can also dissolve and reassemble at a different location (Kolinski and Skolnick 1994).

On the other hand, the move sets in many recent simple lattice model simulations of folding kinetics are limited to local moves (e.g., Abkevich et al. 1994; Pande and Rokhsar 1999). These studies are still useful because they can demonstrate the pos-

sibility of a particular folding path or a folding mechanism, for example, how the conformational search problem can be solved (Abkevich et al. 1994). But obviously they cannot be used to exclude physically plausible folding mechanisms their move set a priori precludes.

Significant move-set dependences have been observed in the folding times of two-dimensional HP model sequences by comparing two move sets, MS1 and MS2 (Chan and Dill 1994). Figure 16.11 shows a similar comparison, now over a continuous class of dynamics models that have different relative weights (probabilities) for the moves in the two move sets. In this particular example, folding times in different dynamics models can differ by more than six orders of magnitude (figure 16.11b). The underlying physical reason is pictorically illustrated in figure 16.11a: a rigid rotation (in MS2)—even at a low diffusion rate (i.e., low relative probability in figure 16.11b)—can bring two assembled fragments of a nonnative conformation together to form the native structure, and can do so without encountering any energy barrier. However, a move set limited to local moves (MS1) must take many steps and break at least two existing HH contacts along the way to arrive at the same end point. This implies that the kinetics with the two move sets are very different when breaking of existing contacts are energetically costly, as is the case in this calculation.

It is clear from this example, as have been pointed out before (Skolnick and Kolinski 1991; Abkevich et al. 1994; Chan and Dill 1994), that rotation and diffusion of intact assembled fragments of a protein, as envisioned by Karplus and Weaver (1976, 1979) in their diffusion-collision-adhesion model, are precluded by lattice move sets with only local moves. Figure 16.11 suggests that different folding mechanisms would be predicted if different move sets are assumed, even though the underlying interaction potential remains unchanged. Therefore, to gain a clear picture of the relationship between protein energetics and the predicted folding mechanisms in a lattice model, the role of move sets must be taken into account.

16.7 Concluding Remarks

We have emphasized physics-based approaches in this survey of computational methods for protein folding, from very brief sketches of all-atom simulations, continuum electrostatics, and high-coordination lattice models to an exposition of simplified statistical mechanics models. This is not a comprehensive review; in fact, we have devoted more effort to raising questions than to providing answers. Nevertheless, it is hoped that the brief summaries, analyses, and cited references in this chapter would make recent developments more accessible to the interested reader, and help

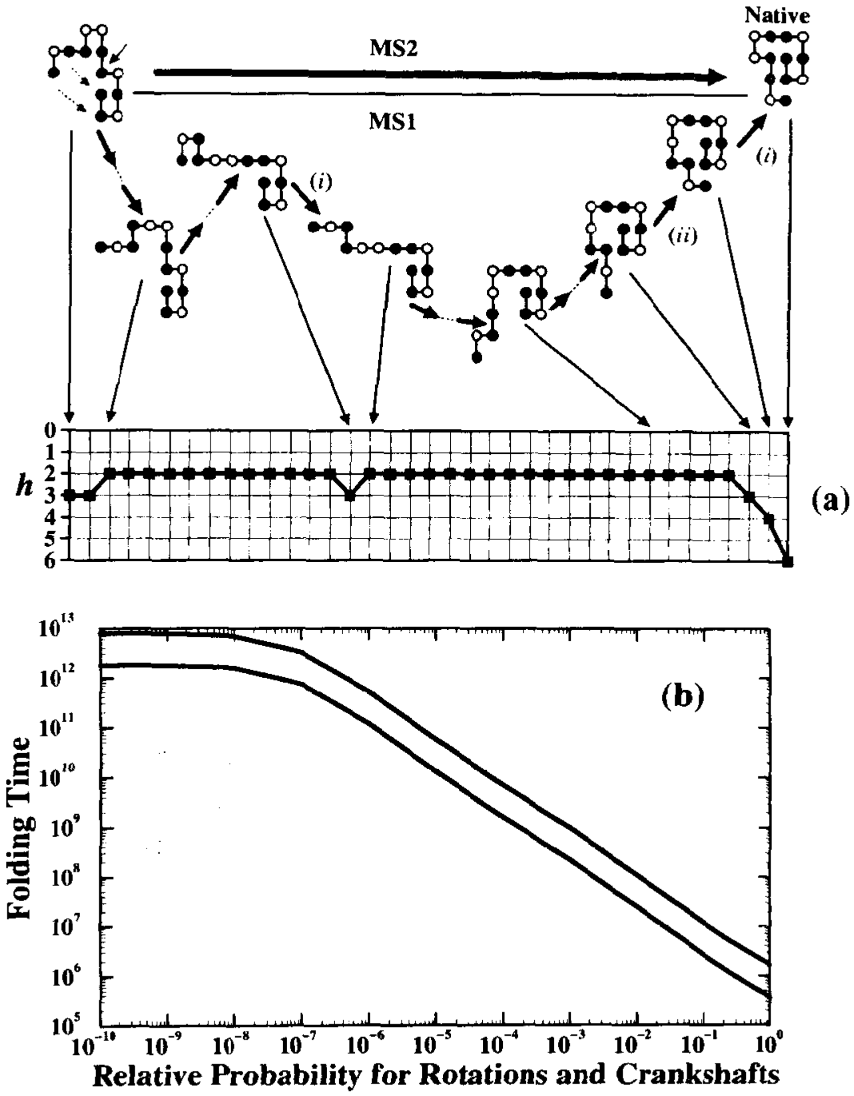


Figure 16.11
Folding kinetics of lattice protein models depend on move set. Two move sets MS1 and MS2 defined by Chan and Dill (1994) are compared using a two-dimensional HP model sequence. MS1 contains only three-point flips and end flips, whereas MS2 includes also crankshafts and rigid rotations. Folding paths from any nonnative conformation to the native conformation (upper right corner) can be very different for the two move sets. For example, the conformation at the upper left corner can reach the native state via one single rigid rotation step in MS2 (top). But at least 36 MS1 steps are needed if the number of energetically unfavorable breaking of HH contacts is minimized along the folding path. (a) is the energy landscape along one such folding path, where h is the number of HH contacts. Selected conformations along this path are shown, chain moves (i) and (ii) are examples of three-point and end flips, respectively. (b) is the folding time of the model HP sequence at HH interaction energy = $-9.3k_BT$, computed by a reduced transition matrix method (Chan and Dill 1994) for a range of probabilities of the additional moves in MS2 (crankshafts and rigid rotations) relative to the local moves in MS1. The upper and lower curves are the time needed to achieve 95 percent and 50 percent native populations, respectively, from an initial ensemble of open conformations. Even relative probabilities as low as 10^{-6} speed up folding considerably in this case.

him or her evaluate results reported in the literature. A few examples have been analyzed and discussed in some detail to highlight issues we believe are important, especially with regard to physical interpretations of simple lattice model results. Protein folding is a vast field; many approaches are complementary to one another. Protein folding is a many faceted problem; its elucidation requires theories at many levels, and models that account for different degrees of complexity. To move forward in physical understanding, it is necessary to critically evaluate theoretical predictions against experiments, to clarify the relationship between the basic modeling assumptions and the physico-chemical forces in real proteins, and to establish rigorous logical connections from a model's basic assumptions to its predictions. Clearly, the degree to which one is successful in these endeavors would determine the amount of physical knowledge and the quality of information that one can gain from any modeling effort.

Acknowledgments

Our research has been supported by Medical Research Council of Canada grant MT-15323, a Premier's Research Excellence Award (Ontario), and a grant from the Connaught Fund to H.S.C., who is a Canada Research Chair in Biochemistry. We thank Erich Bornberg-Bauer, Julie Forman-Kay, and Paul Harrison for their critical reading of the manuscript and helpful suggestions.

References

Abkevich, V. I., Gutin, A. M., and Shakhnovich, E. I. (1994). Specific nucleus as the transition state for protein folding: Evidence from the lattice model. *Biochemistry* 33: 10026–10036.

Abkevich, V. I., Gutin, A. M., and Shakhnovich, E. I. (1995a). Domains in folding of model proteins. *Protein Sci.* 4: 1167–1177.

Abkevich, V. I., Gutin, A. M., and Shakhnovich, E. I. (1995b). Impact of local and non-local interactions on thermodynamics and kinetics of protein folding. *J. Mol. Biol.* 252: 460–471.

Alm, E., and Baker, D. (1999a). Matching theory and experiment in protein folding. *Curr. Opin. Struct. Biol.* 9: 189–196.

Alm, E., and Baker, D. (1999b). Prediction of protein-folding mechanisms from free-energy landscapes derived from native structures. *Proc. Natl. Acad. Sci. USA* 96: 11305–11310.

Alonso, D. O. V., and Daggett, V. (2000). Staphylococcal protein A: Unfolding pathways, unfolded states, and differences between the B and E domains. *Proc. Natl. Acad. Sci. USA* 97: 133–138.

Backofen, R., Will, S., and Bornberg-Bauer, E. (1999). Application of constraint programming techniques for structure prediction of lattice proteins with extended alphabets. *Bioinformatics* 15: 234–242.

Bahar, I., Erman, B., Jernigan, R. L., Atilgan, A. R., and Covell, D. G. (1999). Collective dynamics of HIV-1 reverse transcriptase: Examination of flexibility and enzyme function. *J. Mol. Biol.* 285: 1023–1037.

Baldwin, R. L., and Rose, G. D. (1999). Is protein folding hierarchic? I. Local structure and peptide folding. *Trends Biochem. Sci.* 24: 26–33.

Ben-Tal, N., Sitkoff, D., Topol, I. A., Yang, A.-S., Burt, S. K., and Honig, B. (1997). Free energy of amide hydrogen bond formation in vacuum, in water, and in liquid alkane solution. *J. Phys. Chem. B* 101: 450–457.

Berger, B., and Leighton, T. (1998). Protein folding in the hydrophobic-hydrophilic (HP) model is NP-complete *J. Comp. Biol.* 5: 27–40.

Betancourt, M. R., and Thirumalai, D. (1999a). Pair potentials for protein folding: Choice of reference states and sensitivity of predicted native states to variations in the interaction schemes. *Protein Sci.* 8: 361–369.

Betancourt, M. R., and Thirumalai, D. (1999b). Exploring the kinetic requirements for enhancement of protein folding rates in the GroEL cavity. *J. Mol. Biol.* 287: 627–644.

Borg, J., Jensen, M. H., Sneppen, K., and Tiana, G. (2001). Hydrogen bonds in polymer folding. *Phys. Rev. Lett.* 86: 1031–1033.

Bornberg-Bauer, E. (1997). How are model protein structures distributed in sequence space? *Biophys. J.* 73: 2393–2403.

Bornberg-Bauer, E., and Chan, H. S. (1999). Modeling evolutionary landscapes: Mutational stability, topology, and superfunnels in sequence space. *Proc. Natl. Acad. Sci. USA* 96: 10689–10694.

Bowie, J. U., Reidhaar-Olson, J. F., Lim, W. A., and Sauer, R. T. (1990). Deciphering the message in protein sequences: Tolerance to amino acid substitution. *Science* 247: 1306–1310.

Braxenthaler, M., Unger, R., Auerbach, D., Given, J. A., and Moult, J. (1997). Chaos in protein dynamics. *Proteins: Struct. Funct. Genet.* 29: 417–425.

Broglia, R. A., Tiana, G., Pasquali, S., Roman, H. E., and Vigezzi, E. (1998). *Proc. Natl. Acad. Sci. USA* 95: 12930–12933; erratum: 96: 10943 (1999).

Bryngelson, J. D., Onuchic, J. N., Socci, N. D., and Wolynes, P. G. (1995). Funnels, pathways and the energy landscape of protein folding: A synthesis. *Proteins: Struct. Funct. Genet.* 21: 167–195.

Bryngelson, J. D., and Wolynes, P. G. (1987). Spin glasses and the statistical mechanics of protein folding. *Proc. Natl. Acad. Sci. USA* 84: 7524–7528.

Buchler, N. E. G., and Goldstein, R. A. (2000). Surveying determinants of protein structure designability across different energy models and amino-acid alphabets: A consensus. *J. Chem. Phys.* 112: 2533–2547.

Bussemaker, H. J., Thirumalai, D., and Bhattacharjee, J. K. (1997). Thermodynamic stability of folded proteins against mutations. *Phys. Rev. Lett.* 79: 3530–3533.

Chan, H. S. (1998a). Protein folding: Matching speed and locality. *Nature* 392: 761–763.

Chan, H. S. (1998b). Modeling protein folding by Monte Carlo dynamics: Chevron plots, chevron rollover, and non-Arrhenius kinetics. In *Monte Carlo Approach to Biopolymers and Protein Folding*, Grassberger, P., Barkema, G., and Nadler, W., eds., 29–44, Singapore: World Scientific.

Chan, H. S. (1999). Folding alphabets. *Nature Struct. Biol.* 6: 994–996.

Chan, H. S. (2000). Modeling protein density of states: Additive hydrophobic effects are insufficient for calorimetric two-state cooperativity. *Proteins: Struct. Funct. Genet.* 40: 543–571.

Chan, H. S. (2001). Amino acid side chain hydrophobicity. In *Encyclopedia of Life Sciences*. London: Nature Publishing Group, in press.

Chan, H. S., and Dill, K. A. (1991). Polymer principles in protein structure and stability. *Annu. Rev. Biophys. Biophys. Chem.* 20: 447–490.

Chan, H. S., and Dill, K. A. (1993). The protein folding problem. *Physics Today* 46(2): 24–32.

Chan, H. S., and Dill, K. A. (1994). Transition states and folding dynamics of proteins and heteropolymers. *J. Chem. Phys.* 100: 9238–9257.

Chan, H. S., and Dill, K. A. (1996a). Comparing folding codes for proteins and polymers. *Proteins: Struct. Funct. Genet.* 24: 335–344.

Chan, H. S., and Dill, K. A. (1996b). A simple model of chaperonin-mediated protein folding. *Proteins: Struct. Funct. Genet.* 24: 345–351.

Chan, H. S., and Dill, K. A. (1998). Protein folding in the landscape perspective: Chevron plots and non-Arrhenius kinetics. *Proteins: Struct. Funct. Genet.* 30: 2–33.

Chen, C.-M. (2001). Lattice model of transmembrane polypeptide folding. *Phys. Rev.* E63: 010901(R).

Choi, I. S., Weck, M., Jeon, N. L., and Whitesides, G. M. (2000). Mesoscale folding: A physical realization of an abstract, 2D lattice model for molecular folding. *J. Am. Chem. Soc.* 122: 11997–11998.

Clementi, C., Nymeyer, H., and Onuchic, J. (2000). Topological and energetic factors: What determines the structural details of the transition state ensemble and "en-route" intermediates for protein folding? An investigation for small globular proteins. *J. Mol. Biol.* 298: 937–953.

Cohen, F. E., and Prusiner, S. B. (1998). Pathologic conformations of prion proteins. *Annu. Rev. Biochem.* 67: 793–819.

Cooper, A. (1999). Thermodynamic analysis of biomolecular interactions. *Curr. Opin. Chem. Biol.* 3: 557–563.

Cordes, M. H. J., Burton, R. E., Walsh, N. P., McKnight, C. J., and Sauer, R. T. (2000). An evolutionary bridge to a new protein fold. *Nature Struct. Biol.* 7: 1129–1132.

Cordes, M. H. J., Walsh, N. P., McKnight, C. J., and Sauer, R. T. (1999). Evolution of a protein fold in vitro. *Science* 284: 325–327.

Crescenzi, P., Goldman, D., Papadimitriou, C., Piccolboni, A., and Yannakakis. M. (1998). On the complexity of protein folding. *J. Comp. Biol.* 5: 423–465.

Cui, Y., and Wong, W. H. (2000). Multiple-sequence information provides protection against mis-specified potential energy functions in the lattice model of proteins. *Phys. Rev. Lett.* 85: 5242–5245.

Czaplewski, C., Rodziewicz-Motowidlo, S., Liwo, A., Ripoll, D. R., Wawak, R. J., and Scheraga, H. A. (2000). Molecular simulation study of cooperativity in hydrophobic association. *Protein Sci.* 9: 1235–1245.

Daggett, V. (2000). Long timescale simulations. *Curr. Opin. Struct. Biol.* 10: 160–164.

Daggett, V., and Levitt, M. (1992). A model of the molten globule state from molecular dynamics simulations. *Proc. Natl. Acad. Sci. USA* 89: 5142–5146.

Daggett, V., and Levitt, M. (1994). Protein folding ↔ unfolding dynamics. *Curr. Opin. Struct. Biol.* 4: 291–295.

Daura, X., Jaun, B., Seebach, D., van Gunsteren, W. F., and Mark, A. E. (1998). Reversible peptide folding in solution by molecular dynamics simulation. *J. Mol. Biol.* 280: 925–932.

DeVido, D. R., Dorsey, J. G., Chan, H. S., and Dill, K. A. (1998). Oil/water partitioning has a different thermodynamic signature when the oil solvent chains are aligned than when they are amorphous. *J. Phys. Chem. B* 102: 7272–7279.

Dill, K. A. (1990). Dominant forces in protein folding. *Biochemistry* 29: 7133–7155.

Dill, K. A. (1997). Additivity principles in biochemistry. *J. Biol. Chem.* 272: 701–704.

Dill, K. A., Bromberg, S., Yue, K., Fiebig, K. M., Yee, D. P., Thomas, P. D., and Chan, H. S. (1995). Principles of protein folding—a perspective from simple exact models. *Protein Sci.* 4: 561–602.

Dill, K. A., and Chan, H. S. (1997). From Levinthal to pathways to funnels. *Nature Struct. Biol.* 4: 10–19.

Dill, K. A., and Stigter, D. (1995). Modeling protein stability as heteropolymer collapse. *Adv. Protein Chem.* 47: 59–104.

Dinner, A. R., and Karplus, M. (1998). A metastable state in folding simulations of a protein model. *Nature Struct. Biol.* 5: 236–241.

Dobson, C. M. (1999). Protein misfolding, evolution and disease. *Trends Biochem. Sci.* 24: 329–332.

Doruker, P., Atilgan, A. R., and Bahar, I. (2000). Dynamics of proteins predicted by molecular dynamics simulations and analytical approaches: Application to α-amylase. *Proteins: Struct. Funct. Genet.* 40: 512–524.

Duan, Y., and Kollman, P. A. (1998). Pathways to a protein folding intermediate observed in a 1-microsecond simulation in aqueous solution. *Science* 282: 740–744.

Edsall, J. T. (1995). Hsien Wu and the first theory of protein denaturation (1931). *Adv. Protein Chem.* 46: 1–5.

Fauchère, J.-L., and Pliška, V. (1983). Hydrophobic parameters Π of amino-acid side chains from the partitioning of *N*-acetyl-amino-acid amides. *Eur. J. Med. Chem.—Chim. Ther.* 18: 369–375.

Feldman, H. J., and Hogue, C. W. V. (2000). A fast method to sample real protein conformational space. *Proteins: Struct. Funct. Genet.* 39: 112–131.

Finkelstein, A. V. (1998). 3D protein folds: Homologs against errors—a simple estimate based on the random energy model. *Phys. Rev. Lett.* 80: 4823–4825.

Fontana, W., and Schuster, P. (1987). A computer model of evolutionary optimization. *Biophys. Chem.* 26: 123–147.

Fontana, W., and Schuster, P. (1998). Contintuity in evolution: On the nature of transitions. *Science* 280: 1451–1455.

Forman-Kay, J. D. (1999). The "dynamics" in the thermodynamics of binding. *Nature Struct. Biol.* 6: 1086–1087.

Freire, E. (1995). Differential scanning calorimetry. In *Methods in Molecular Biology*, vol. 40, *Protein Stability and Folding: Theory and Practice*, Shirley, B. A. ed., 191–218. Totowa, NJ: Humana Press Inc.

Galzitskaya, O. V., and Finkelstein, A. V. (1999). A theoretical search for folding/unfolding nuclei in three-dimensional protein structures. *Proc. Natl. Acad. Sci. USA* 96: 11299–11304.

Gilson, M. K. (1995). Theory of electrostatic interactions in macromolecules. *Curr. Opin. Struct. Biol.* 5: 216–223.

Giugliarelli, G., Micheletti, C., Banavar, J. R., and Maritan, A. (2000). Compactness, aggregation and prion-like behavior of protein—a lattice model study. *J. Chem. Phys.* 113: 5072–5077.

Gō, N. (1983). Theoretical studies of protein folding. *Annu. Rev. Biophys. Bioeng.* 12: 183–210.

Godzik, A., Kolinski, A., and Skolnick, J. (1993). Lattice representations of globular proteins: How good are they? *J. Comp. Chem.* 14: 1194–1202.

Godzik, A., Kolinski, A., and Skolnick, J. (1995). Are proteins ideal mixtures of amino acids? Analysis of energy parameter sets. *Protein Sci.* 4: 2107–2117.

Govindarajan, S., and Goldstein, R. A. (1997). Evolution of model proteins on a foldability landscape. *Proteins: Struct. Funct. Genet.* 29: 461–466.

Gupta, P., Hall, C. K., and Voegler, A. C. (1998). Effect of denaturant and protein concentrations upon protein refolding and aggregation: A simple lattice model. *Protein Sci.* 7: 2642–2652.

Gutin, A. M., Abkevich, V. I., and Shakhnovich, E. I. (1998). A protein engineering analysis of the transition state for protein folding: Simulation in the lattice model. *Fold. Des.* 3: 183–194.

Haliloglu, T., and Bahar, I. (1999). Structure-based analysis of protein dynamics near the folded state. Results for hen lysozyme, and comparison with X-ray diffraction and NMR relaxation data. *Proteins: Struct. Funct. Genet.* 37: 654–667.

Hansmann, U. H. E., and Okamoto, Y. (1999). New Monte Carlo algorithms for protein folding. *Curr. Opin. Struct. Biol.* 9: 177–183.

Hao, M.-H., and Scheraga, H. A. (1998). Molecular mechanisms for cooperative folding of proteins. *J. Mol. Biol.* 277: 973–983.

Harrison, P. M., Chan, H. S., Prusiner, S. B., and Cohen, F. E. (1999). Thermodynamics of model prions and its implications for the problem of prion protein folding. *J. Mol. Biol.* 286: 593–606.

Harrison, P. M., Chan, H. S., Prusiner, S. B., and Cohen, F. E. (2001). Conformational propagation with prion-like characteristics in a simple model of protein folding. *Protein Sci.* 10: 819–835.

Hinds, D. A., and Levitt, M. (1996). From structure to sequence and back again. *J. Mol. Biol.* 258: 201–209.

Hirst, J. D. (1999). The evolutionary landscape of functional model proteins. *Protein Eng.* 12: 721–726.

Honig, B., and Nicholls, A. (1995). Classical electrostatics in biology and chemistry. *Science* 268: 1144–1149.

Honig, B., and Yang, A.-S. (1995). Free energy balance in protein folding. *Adv. Protein Chem.* 46: 27–58.

Hummer, G., Garde, S., García, A. E., and Pratt, L. R. (2000). New perspectives on hydrophobic effects. *Chem. Phys.* 258: 349–370.

Irbäck, A., Peterson, C., and Potthast, F. (1996). Evidence for nonrandom hydrophobicity structures in protein chains. *Proc. Natl. Acad. Sci. USA* 93: 9533–9538.

Irbäck, A., Peterson, C., Potthast, F., and Sandelin, E. (1998). Monte Carlo procedure for protein design. *Phys. Rev. E* 58: R5249–R5252.

Irbäck, A., Peterson, C., Potthast, F., and Sommelius, O. (1997). Local interactions and protein folding: A 3D off-lattice approach. *J. Chem. Phys.* 107: 273–282.

Irbäck, A., and Sandelin, E. (2000). On hydrophobicity correlations in protein chains. *Biophys. J.* 79: 2252–2258.

Irbäck, A., Sjunnesson, F., and Wallin, S. (2000). Three-helix-bundle protein in a Ramachandran model. *Proc. Natl. Acad. Sci. USA* 97: 13614–13618.

Istrail, S., Schwartz, R., and King, J. (1999). Lattice simulations of aggregation funnels for protein folding. *J. Comp. Biol.* 6: 143–162.

Jackson, J. D. (1975). *Classical Electrodynamics*, 2nd ed., New York: John Wiley & Sons, Inc.

Jernigan, R. L., and Bahar, I. (1996). Structure-derived potentials and protein simulations. *Curr. Opin. Struct. Biol.* 6: 195–209.

Karplus, M., and Weaver, D. L. (1976). Protein-folding dynamics. *Nature* 160: 404–406.

Karplus, M., and Weaver, D. L. (1979). Diffusion-collision model for protein folding. *Biopolymers* 18: 1421–1427.

Karplus, P. A. (1997). Hydrophobicity regained. *Protein Sci.* 6: 1302–1307.

Kauzmann, W. (1948). The nature of glassy state and the behavior of liquids at low temperatures. *Chem. Rev.* 43: 219–256.

Kauzmann, W. (1959). Some factors in the interpretation of protein denaturation. *Adv. Protein Chem.* 14: 1–63.

Kay, L. E. (1998). Protein dynamics from NMR. *Biochem. Cell. Biol.* 76: 145–152.

Kaya, H., and Chan, H. S. (2000a). Polymer principles of protein calorimetric two-state cooperativity. *Proteins: Struct. Funct. Genet.* 40: 637–661. Erratum: 43: 523 (2001).

Kaya, H., and Chan, H. S. (2000b). Energetic components of cooperative protein folding. *Phys. Rev. Lett.* 85: 4823–4826.

Kelly, J. W. (1998). The alternative conformations of amyloidogenic proteins and their multi-step assembly pathways. *Curr. Opin. Struct. Biol.* 8: 101–106.

Klimov, D. K., and Thirumalai, D. (2000). Mechanisms and kinetics of beta-hairpin formation. *Proc. Natl. Acad. Sci. USA* 97: 2544–2549.

Kolinski, A., Godzik, A., and Skolnick, J. (1993). A general method for the prediction of the three dimensional structure and folding pathway of globular proteins: Application to designed helical proteins. *J. Chem. Phys.* 98: 7420–7433.

Kolinski, A., Ilkowski, B., and Skolnick, J. (1999). Dynamics and thermodynamics of β-hairpin assembly: Insights from various simulation techniques. *Biophys. J.* 77: 2942–2952.

Kolinski, A., and Skolnick, J. (1994). Monte Carlo simulations of protein folding. I. Lattice model and interaction scheme. *Proteins: Struct. Funct. Genet.* 18: 338–352.

Kolinski, A., Skolnick, J., and Yaris, R. (1987). Monte Carlo studies on equilibrium globular protein folding. I. Homopolymeric lattice models of β-barrel proteins. *Biopolymers* 26: 937–962.

Lau, K. F., and Dill, K. A. (1989). Theory for protein mutability and biogenesis. *Proc. Natl. Acad. Sci. USA* 87: 638–642.

Laughlin, R. B., Pines, D., Schmalian, J., Stojković, B. P., and Wolynes, P. (2000). The middle way. *Proc. Natl. Acad. Sci. USA* 97: 32–37.

Lazaridis, T., and Karplus, M. (1997). "New view" of protein folding reconciled with the old through multiple unfolding simulations. *Science* 278: 1928–1931.

Leach, A. R. (1996). *Molecular Modelling: Principles and Applications.* Singapore: Longman.

Leopold, P. E., Montal, M., and Onuchic, J. N. (1992). Protein folding funnels: A kinetic approach to the sequence-structure relationship. *Proc. Natl. Acad. Sci. USA* 89: 8721–8725.

Li, H., Helling, R., Tang, C., and Wingreen, N. (1996). Emergence of preferred structures in a simple model of protein folding. *Science* 273: 666–669.

Li, H., Tang, C., and Wingreen, N. S. (1997). Nature of driving force for protein folding: A result from analyzing the statistical potential. *Phys. Rev. Lett.* 79: 765–768.

Li, L. M., and Lindquist, S. (2000). Creating a protein-based element of inheritance. *Science* 287: 661 664.

Lindquist, S. (1997). Mad cows meet psi-chotic yeast: The expansion of the prion hypothesis. *Cell* 89: 495–498.

Lipman, D. J., and Wilbur, W. J. (1991). Modelling neutral and selective evolution of protein folding. *Proc. R. Soc. London B* 245: 7–11.

Lorimer, G. (1997). Protein folding—Folding with a two-stroke motor. *Nature* 388: 720–723.

Makhatadze, G. I., and Privalov, P. L. (1995). Energetics of protein structure. *Adv. Protein Chem.* 47: 307–425.

Mark, A. E., and van Gunsteren, W. F. (1994). Decomposition of the free-energy of a system in terms of specific interactions—implications for theoretical and experimental studies. *J. Mol. Biol.* 240: 167–176.

Micheletti, C., Banavar, J., Maritan, A., and Seno, F. (1999). Protein structures and optimal folding emerging from a geometrical variational principle. *Phys. Rev. Lett.* 82: 3372–3375.

Micheletti, C., Seno, F., Maritan, A., and Banavar, J. R. (1998). Design of proteins with hydrophobic and polar amino acids. *Proteins: Struct. Funct. Genet.* 32: 80–87.

Miller, R., Danko, C. A., Fasolka, J., Balazs, A. C., Chan, H. S., and Dill, K. A. (1992). Folding kinetics of proteins and copolymers. *J. Chem. Phys.* 96: 768–780.

Mirny, L. A., Abkevich, V. I., and Shakhnovich, E. I. (1998). How evolution makes proteins fold quickly. *Proc. Natl. Acad. Sci. USA* 95: 4976–4981.

Mirny, L. A., and Shakhnovich, E. I. (1996). How to derive a protein folding potential? A new approach to an old problem. *J. Mol. Biol.* 264: 1164–1179.

Miyazawa, S., and Jernigan, R. L. (1985). Estimation of effective inter-residue contact energies from protein crystal structures: Quasi-chemical approximation. *Macromolecules* 18: 534–552.

Miyazawa, S., and Jernigan, R. L. (1996). Residue-residue potentials with a favorable contact pair term and an unfavorable high packing density term, for simulation and threading. *J. Mol. Biol.* 256: 623–644.

Moult, J., Hubbard, T., Fidelis, K., and Pedersen, J. T. (1999). Critical assessment of methods of protein structure prediction (CASP): Round III. *Proteins: Struct. Funct. Genet.* suppl. 3: 2–6.

Muñoz, V., and Eaton, W. A. (1999). A simple model for calculating the kinetics of protein folding from three-dimensional structures. *Proc. Natl. Acad. Sci. USA* 96: 11311–11316.

Myers, J. K., and Pace, C. N. (1996). Hydrogen bonding stabilizes globular proteins. *Biophys. J.* 71: 2033 2039.

Nelson, E. D., and Onuchic, J. N. (1998). Proposed mechanism for stability of proteins to evolutionary mutations. *Proc. Natl. Acad. Sci. USA* 95: 10682–10686.

Onuchic, J. N., Luthey-Schulten, Z., and Wolynes, P. G. (1997). Theory of protein folding: The energy landscape perspective. *Annu. Rev. Phys. Chem.* 48: 545–600.

Osguthorpe, D. J. (2000). Ab initio protein folding. *Curr. Opin. Struct. Biol.* 2: 146–152.

Pace, C. N., Hebert, E. J., Shaw, K. L., Schell, D., Both, V., Krajcikova, D., Sevcik, J., Wilson, K. S., Dauter, Z., Hartley, R. W., and Grimsley, G. R. (1998). Conformational stability and thermodynamics of folding of ribonucleases Sa, Sa2 and Sa3. *J. Mol. Biol.* 279: 271–286.

Pande, V. S., Grosberg, A. Yu., and Tanaka, T. (1997). Statistical mechanics of simple models of protein folding and design. *Biophys. J.* 73: 3192–3210.

Pande, V. S., and Rokhsar, D. S. (1999). Folding pathway of a lattice model for proteins. *Proc. Natl. Acad. Sci. USA* 96: 1273–1278.

Park, B. H., and Levitt, M. (1995). The complexity and accuracy of discrete state models of protein structure. *J. Mol. Biol.* 249: 493–507.

Pathria, R. K. (1980). *Statistical Mechanics*, Oxford, U.K.: Pergamon Press.

Perelson, A. S., and Macken, C. A. (1995). Protein evolution on partially correlated landscapes. *Proc. Natl. Acad. Sci. USA* 92: 9657–9661.

Plaxco, K. W., Simons, K. T., and Baker, D. (1998). Contact order, transition state placement and the refolding rates of single domain proteins. *J. Mol. Biol.* 277: 985–994.

Plaxco, K. W., Simons, K. T., Ruczinski, I., and Baker, D. (2000). Topology, stability, sequence, and length: Defining the determinants of two-state protein folding kinetics. *Biochemistry* 39: 11177–11183.

Privalov, P. L. (1979). Stability of proteins. Small globular proteins. *Adv. Protein Chem.* 33: 167–241.

Rank, J. A., and Baker, D. (1997). A desolvation barrier to hydrophobic cluster formation may contribute to the rate-limiting step in protein folding. *Protein Sci.* 6: 347–354.

Rapaport, D. C. (1997). *The Art of Molecular Dynamics Simulation.* New York: Cambridge University Press.

Roux, B., and Simonson, T. (1999). Implicit solvent models. *Biophys. Chem.* 78: 1-20.

Šali, A., and Kuriyan, J. (1999). Challenges at the frontiers of structural biology. *Trends Biochem. Sci.* 24: M20–M24.

Sfatos, C. D., Gutin, A. M., Abkevich, V. I., and Shakhnovich, E. I. (1996). Simulation of chaperone-assisted folding. *Biochemistry* 35: 334–339.

Shakhnovich, E. I. (1994). Protein with selected sequence fold into unique native conformation. *Phys. Rev. Lett.* 72: 3907–3910; Erratum: 74: 2618 (1995).

Shakhnovich, E. I. (1997). Theoretical studies of protein-folding thermodynamics and kinetics. *Curr. Opin. Struct. Biol.* 7: 29–40.

Shakhnovich, E. I. (1998). Folding nucleus: Specific or multiple? Insights from lattice models and experiments. *Fold. Des.* 3: R108–R111.

Shakhnovich, E., Abkevich, V., and Ptitsyn, O. (1996). Conserved residues and the mechanism of protein folding. *Nature* 379: 96–98.

Shakhnovich, E. I., Farztdinov, G., Gutin, A. M., and Karplus, M. (1991). Protein folding bottlenecks: A lattice Monte Carlo simulation. *Phys. Rev. Lett.* 67: 1665–1668.

Shimizu, S., and Chan, H. S. (2000). Temperature dependence of hydrophobic interactions: A mean force perspective, effects of water density, and non-additivity of thermodynamic signatures. *J. Chem. Phys.* 113: 4683–4700.

Shimizu, S., and Chan, H. S. (2001a). Configuration-dependent heat capacity of pairwise hydrophobic interactions. *J. Am. Chem. Soc.* 123: 2083–2084.

Shimizu, S., and Chan, H. S. (2001b). Anti-cooperativity in hydrophobic interactions: A simulation study of spatial dependence of three-body effects and beyond. *J. Chem. Phys.* 115: 1414–1421.

Shimizu, S., and Chan, H. S. (2001c). Statistical mechanics of solvophobic aggregation: Additive and cooperative effects. *J. Chem. Phys.* 115: 3424–3431.

Shortle, D. (1996). The denatured state (the other half of the folding equation) and its role in protein stability. *FASEB J.* 10: 27–34.

Shortle, D. (2000). Prediction of protein structure. *Curr. Biol.* 10: R49–R51.

Sigler, P. B., Xu, A., Rye, H. S., Burston, S. G., Fenton, W. A., and Horwich, A. L. (1998). Structure and function in GroEL-mediated protein folding. *Annu. Rev. Biochem.* 67: 581–608.

Sikorski, A., Kolinski, A., and Skolnick, J. (2000). Computer simulations of the properties of the α_2, α_2C, and α_2D de novo designed helical proteins. *Proteins: Struct. Funct. Genet.* 38: 17–28.

Sippl, M. J. (1995). Knowledge-based potentials for proteins. *Curr. Opin. Struct. Biol.* 5: 229–235.

Skolnick, J., and Kolinski, A. (1991). Dynamic Monte Carlo simulations of a new lattice model of globular protein folding, structure and dynamics. *J. Mol. Biol.* 221: 449–531.

Sorenson, J. M., Hura, G., Soper, A. K., Pertsemlidis, A., and Head-Gordon, T. (1999). Determining the role of hydration forces in protein folding. *J. Phys. Chem. B* 103: 5413–5426.

Spector, S., and Raleigh, D. P. (1999). Submillisecond folding of the peripheral subunit-binding domain. *J. Mol. Biol.* 293: 763–768.

Sun, S. (1993). Reduced representation model of protein structure prediction: Statistical potential and genetic algorithms. *Protein Sci.* 2: 762–785.

Sun, S., Thomas, P. D., and Dill, K. A. (1995). A simple protein folding algorithm using a binary code and secondary structure constraints. *Protein Eng.* 8: 769–778.

Taketomi, H., Ueda, Y., and Gō, N. (1975). Studies on protein folding, unfolding and fluctuations by computer simulation. 1. The effect of specific amino acid sequence represented by specific inter-unit interactions. *Int. J. Pept. Protein Res.* 7: 445–459.

Tanaka, S., and Scheraga, H. A. (1976). Medium- and long-range interaction parameters between amino acids for predicting three-dimensional structures of proteins. *Macromolecules* 9: 945–950.

Tang, C. (2000). Simple models of the protein folding problem. *Physica A* 288: 31–48.

Thirumalai, D., and Klimov, D. K. (1998) Fishing for folding nuclei in lattice models and proteins. *Fold. Des.* 3: R112–R118.

Thirumalai, D., and Klimov, D. K. (1999). Deciphering the timescales and mechanisms of protein folding using minimal off-lattice models. *Curr. Opin. Struct. Biol.* 9: 197–207.

Thirumalai, D., and Woodson, S. A. (1996). Kinetics of folding of proteins and RNA. *Acc. Chem. Res.* 29: 433–439.

Thomas, P. D., and Dill, K. A. (1996). Statistical potentials extracted from protein structures: How accurate are they? *J. Mol. Biol.* 257: 457–469.

Todd, M. J., Lorimer, G. H., and Thirumalai, D. (1996). Chaperonin-facilitated protein folding: Optimization of rate and yield by an iterative annealing mechanism. *Proc. Natl. Acad. Sci. USA* 93: 4030–4035.

Van der Hoef, M. A., and Madden, P. A. (1999). Three-body dispersion contributions to the thermodynamic properties and effective pair interactions in liquid argon. *J. Chem. Phys.* 111: 1520–1526.

Van Nimwegen, E., Crutchfield, J. P., and Huynen, M. (1999). Neutral evolution of mutational robustness. *Proc. Natl. Acad. Sci. USA* 96: 9716–9720.

Vendruscolo, M., Najmanovich, R., and Domany, E. (2000). Can a pairwise contact potential stabilize native protein folds against decoys obtained by threading? *Proteins: Struct. Funct. Genet.* 38: 134–148.

Voigt, C. A., Kauffman, S., and Wang, Z.-G. (2001). Rational evolutionary design: The theory of in vitro protein evolution. *Adv. Protein Chem.* 55: 79–160.

Wang, J., and Wang, W. (1999). A computational approach to simplifying the protein folding alphabet. *Nature Struct. Biol.* 6: 1033–1038.

Wang, J., and Wang, W. (2000). Modeling study on the validity of a possibly simplified representation of proteins. *Phys. Rev. E* 61: 6981–6986.

Wolynes, P. G. (1997). As simple as can be? *Nature Struct. Biol.* 4: 871–874.

Wu, H. (1931). Studies on denaturation of proteins XIII. A theory of denaturation. *Chinese J. Physiol.* 5: 321–344 (Reprinted in *Adv. Protein Chem.* 46: 6–26 [1995]).

Yue, K., and Dill, K. A. (1995). Forces of tertiary structural organization in globular proteins. *Proc. Natl. Acad. Sci. USA* 92: 146–150.

Yue, K., and Dill, K. A. (1996). Folding proteins with a simple energy function and extensive conformational searching. *Protein Sci.* 5: 254–261.

Yue, K., Fiebig, K. M., Thomas, P. D., Chan, H. S., Shakhnovich, E. I., and Dill, K. A. (1995). A test of lattice protein folding algorithms. *Proc. Natl. Acad. Sci. USA* 92: 325–329.

Zhang, L., and Skolnick, J. (1998). How do potentials derived from structural databases relate to "true" potentials? *Protein Sci.* 7: 112–122.

Zhou, Y. Q., and Karplus, M. (1999). Interpreting the folding kinetics of helical proteins. *Nature* 401: 400 403.

Zidek, L., Novotny, M. V., and Stone, M. J. (1999). Increased protein backbone conformational entropy upon hydrophobic ligand binding. *Nature Struct. Biol.* 6: 1118–1121.

17 Protein Structure Prediction by Comparison: Homology-Based Modeling

Manuel C. Peitsch, Torsten Schwede, Alexander Diemand, and Nicolas Guex

17.1 Introduction

Understanding the function and the physiological role of proteins is a basic requirement for the discovery of novel medicines (small molecules) and "biologicals" (protein-based products) with medical, industrial, or commodity applications. Although the draft sequence of the complete human genome is about to be ready, humankind is very far from understanding the function and the physiological role of the gene products it encodes. Indeed, being able to read the letters and the words is disconnected from understanding their meaning. Therefore, the attention of many biologists is now shifting to the functional analysis of the genome. The term "functional analysis," however, merits a few short comments, as one can easily distinguish three levels of gene-related knowledge. First, once a gene sequence is known, one has to discover what transcripts it encodes. Indeed, genes can have several splice products yielding as many raw protein transcripts. Then, each transcript can be further altered by post-transcriptional modifications, which can be context specific (such as cell cycle). Second, proteins will often be attributed to a broad functional class (such as kinase, G protein-coupled receptor, etc.) by analogy with well-studied family members. However, in most cases, it is necessary to confirm these inferred functions through experimental approaches. By doing so, the knowledge base on protein will grow and functional predictions will become increasingly precise. Third, the physiological role of a protein has to be defined, meaning that the precise role of the protein in physiological processes has to be uncovered. This is a higher level of knowledge than the function. Indeed, one can know the precise activity of an enzyme, including well worked-out reaction kinetics, but not understand how it contributes to the processes of life. This is very frequent, as one still discovers new roles for well-studied proteins. The "full picture" will thus take a very long time before it takes shape and represents the ultimate grand challenge in biology.

Functional analysis, the first major step after genome sequencing and gene identification, must rely on a combination of technologies. Consequently, new experimental approaches, and their automation for large-scale applications, will need development. Concurrently, and in order to maximize the value of large data sets, one will witness the development of new data mining methods and mathematical models for biological processes simulation.

A protein's function is tightly linked to its three-dimensional (3D) structure. As residues located far apart in the primary sequence can be very close in space, and only a few residues are generally responsible for a protein's function, insights into the

3D structure of a protein can represent a key component of the functional analysis process. Consequently, an atomic-level 3D representation to assign roles to specific residues is a major asset, both for planning experiments and explaining observations.

The "folding" process of a protein is very complex and as yet there is no objective and reliable way to determine it from the sole sequence. The scientific community is thus dependent on experimental protein structure elucidation. The usual approaches, both X-ray diffraction and NMR (nuclear magnetic resonance), are both hampered by many technical hurdles and limitations. Consequently, several concerted "structural genomics" efforts are being launched in both private and the public sector to address these difficulties and increase the throughput of experimental structure elucidation. However, these will not be sufficient to elucidate the structure of all proteins of interest. Although today the protein sequence databases SWISS-PROT and trEMBL are populated with over 250,000 proteins, only approximately 10,000 of them have known 3D structures. Furthermore, SWISS-PROT and trEMBL hold fewer than 20,000 human proteins, meaning that at least another 80,000 will be added in the near future. The direct consequences of this is that only the "highly interesting" proteins should be elucidated experimentally in the foreseeable future.

In this context, comparative modeling methods (homology based) have been developed and have matured to a point where many of the resulting models yield enough insights into a protein's 3D structure to be useful in functional analysis (Westhead and Thornton 1998).

17.2 What Is Comparative Protein Modeling?

Proteins from different sources and sometimes diverse biological functions can have similar sequences. It is generally accepted that high sequence similarity is reflected by structural similarity. Indeed, the relative mean square deviation (rmsd) of the alpha-carbon co-ordinates for protein cores sharing 50 percent residue identity is expected to be around 1 Å (Chothia and Lesk 1986). Thus the most reliable prediction methods, termed comparative protein modeling (also often called modeling by homology), consist of the extrapolation of the structure for a new (target) sequence from the known 3D structure of related family members (templates) (Bajorath et al. 1993).

17.2.1 Identification of Modeling Templates

Comparative protein modeling requires at least one sequence of known 3D structure with significant similarity to the target sequence. In order to determine if a modeling request can be carried out, one generally compares the target sequence with a data-

base of sequences derived from the Brookhaven Protein Data Bank (PDB) (Bernstein et al. 1977). This can be performed using sequence similarity search tools such as FastA (Pearson and Lipman 1988), BLAST (Altschul et al. 1997), and PSI-BLAST (Altschul et al. 1997). Generally speaking, sequences with a FastA score 10.0 standard deviations above the mean of the random scores and a Poisson unlikelyhood probability $P(N)$ lower than 10^{-5} (BLAST) can be considered for the model building procedure. The choice of template structures should be further restricted to those that share at least 25 percent residue identity with 40 percent of the target sequence as determined by sequence alignment algorithms such as SIM (Huang and Miller 1991).

The above procedure might allow the selection of several suitable templates for a given target sequence. The best template structure—the one with the highest sequence similarity to the target—should serve as the *reference*. Other selected templates can be used and must be optimally superposed onto it in 3D. As a result, each residue of the reference structure is then aligned with a residue from the other available templates. This yields a structurally corrected multiple sequence alignment.

17.2.2 Aligning the Target Sequence with the Template Sequence(s)

The target sequence can then be aligned with the template sequence or, if several templates are selected, with the structurally corrected multiple sequence alignment, using the best-scoring diagonals obtained by sequence alignment algorithms such as SIM (Huang and Miller 1991). Residues, which should not be used for model building, for example those located in non-conserved loops, will be ignored during the modeling process. Thus, the common core of the target protein and the loops completely defined by at least one supplied template structure will be built.

17.2.3 Building the Model

Two very distinct classes of methods have been developed to build a model. One, implemented in MODELLER (Sali and Blundell 1993), is based on the satisfaction of spatial restraints derived from the alignment between the target sequence and its templates. The other, described below, is based on the building of an averaged framework from the coordinates of the templates.

The coordinates of the model can then be built. Initially, an average framework for the target sequence will by calculated based on the multiple sequence alignment described above. The spatial location for as many atoms of the target as possible will be derived from the positions of the corresponding atoms in the template structures. The coordinates will be calculated as a weight averaged position based on the corresponding atoms available from the templates. Each template will contribute to the

average to an extent determined by its local degree of sequence similarity with the target (Peitsch 1995, 1996). As the selected templates do not contain structural information about non-conserved loops and many of the side chains, these must be rebuilt in consecutive steps.

Rebuilding non-conserved loops can be performed using a "spare parts" algorithm, as described by Jones and Thirup (Jones and Thirup 1986; Greer 1991). Although most of the known 3D structures available share no sequence or structure similarity with the target and templates, there might be similarities in the loop regions, which can be inserted in the protein model. Each loop is defined by its length and the geometry of its "stems," namely the coordinates of the alpha carbon (C^{α}) atom of the four residues preceding and following the loop. The fragments that correspond to the above loop definition are extracted from the PDB entries if the rmsd computed for their "stems" is lower than a specified cut-off value. Furthermore, only fragments that do not overlap with neighboring parts of the structure are considered possible candidates. The accepted spare parts are sorted according to their rmsd and their degree of sequence similarity with the target. The best fitting fragment is then added to the model.

Because the spare parts algorithm does not always lead to convincing solutions, one can also use an approach based on a conformational space search driven by the satisfaction of stereochemical, distance, and steric constraints. Loops modeled with these methods are filtered according to criteria such as the surface exposure of hydrophobic moieties and relative conformational energies.

In the final step of coordinate generation, it is necessary to determine the geometry of fully defined side chains, and build those that are incomplete or lacking. This can be achieved by using libraries of allowed side chain rotamers. Such tables are deduced from the highest resolution structures from the PDB, and each rotamer is ranked according to its frequency of occurrence (Ponder and Richards 1987). More sophisticated tables provide the distribution of the side chain rotamers based on backbone conformations (Helix, Sheet, Turn, and Coil). Whereas the optimization of the fully defined side chain geometries is performed by replacing each one by its best fitting rotamer, the partially defined and lacking residues are built by searching for a combination of side chain conformations that minimizes the steric overlaps in the model structure.

17.2.4 Model Refinement

The final step of the coordinate generation process in the idealization of stereochemistry of the model; it consists mainly of the optimization of bond geometry and

the removal of unfavorable non-bonded contacts. This step can be performed by energy minimization packages such as CHARMM (Brooks et al. 1983) or GROMOS (van Gunsteren et al. 1996).

Excessive energy minimization will cause the model to deviate markedly from the original model, which is not suitable and should be avoided. Indeed, experience has shown that the changes induced by force field computations do not improve the accuracy of the model with respect to a control experimental structure (Moult 1999; Venclovas et al. 1999). Thus, one should keep the number of minimization cycles to a minimum, but sufficient to improve the stereochemistry of the model. A typical energy minimization procedure will use no more than three hundred cycles of energy minimization (a combination of 50–100 steps of steepest descent and 200–250 steps of conjugate gradient minimization) while imposing a harmonic force constant to the C^{α} carbon atoms.

17.3 Automated Protein Modeling

By comparing all entries of the SWISS-PROT/trEMBL sequence database (release 36) with a non-redundant subset of all sequences of known 3D structure, we found that approximately 30 percent of the database entries have at least one suitable modeling template for a portion of their sequence. This number increases every year by roughly 5 percent, showing that the growth in available templates allows an ever increasing number of sequences to be modeled by comparative methods. Hence, we might speculate that comparative modeling procedures could be applied to more than 90 percent of the protein sequences within the next 20 years. However, this projection does not address the issue that many protein structures do not cover the full length of the transcripts. Indeed, it is generally necessary to express only a domain or part of a protein in order to get crystals, so one often has only partial structural information for a protein sequence. A typical example is the Fas ligand, which is composed of three domains: first, an intracellular domain for which there is an NMR-derived structure (PDB entry 1DDF); second, a membrane spanning domain for which there is no experimental information; and third, although there is no available experimental structure for the extracellular domain, a model can easily be built based on the tumor necrosis factor structure (PDB entry 1TNR). This means that in many cases, especially for multidomain proteins, several structure elucidation experiments have to be conducted before the entire sequence can be covered with structural information, and even then the overall structure will only result from an assembly of "pieces." Furthermore, the N-terminal and C-terminal parts of many protein structures are not

resolved well enough and thus cannot be "seen" in the electron density map. Consequently, the average length of a protein model will be of 180 residues (ranging from 30 to 900).

For the sequences that share a similarity of at least 30–35 percent with a template, it is generally feasible to attempt comparative protein modeling using a completely automated approach with some limitations, which we will illustrate later in this chapter. The emerging structural genomics efforts will make a major contribution to the broader application of comparative modeling as they will yield new modeling templates at a high rate. However, most large multidomain complex models cannot presently be built, mainly because no suitable modeling templates are available for such complexes and because the prediction of protein-protein contacts is inaccurate and still in its infancy. There is no doubt that the scrutiny of multiple sequence alignments is more informative than the analysis of an isolated sequence. This allows the identification of conserved and variable residues, and improves the rationalization of site-directed mutagenesis experiments. Likewise, comparative structure analysis involving several members of a protein family, as opposed to single structure inspection, is expected to be invaluable in a number of situations. The understanding of the structural differences between residues occupying a similar portion of space in several family members or species variants will provide a strong basis for such applications as planning mutagenesis experiments and understanding the selectivity of compounds binding to active sites (drug design). Such a collection of structures can also be used during the rational design of combinatorial libraries of compound. It will thus be increasingly valuable to have models for as many members of a protein family as possible.

17.4 Very Large-Scale Protein Modeling

Two years ago, we submitted all entries of the sequence databases SWISS-PROT and trEMBL to comparative protein modeling in a single experiment (Guex et al. 1999), called 3DCrunch. This followed several smaller experiments during which the full protein complement of bacterial and the yeast genome we (Peitsch 1997; Peitsch et al. 1997) and others (Sanchez and Sali 1998) subjected to comparative modeling. In all cases we used the software framework underlying SWISS-MODEL. For this project, run in collaboration with Silicon Graphics Inc., we used a 64-processor Silicon Graphics CRAY Origin2000 server with 32Gb of memory. 3DCrunch needed 6828.4 cpu-hours to complete (4.4 days). This corresponds to 284.5 days of computing on a single processor machine. During the experiment, over 64,000 protein models

were generated. This project is ongoing, as both the Protein Data Bank and SWISS-PROT/trEMBL are updated regularly and thus new sequences can be modeled on a monthly basis. To date, over 90,000 protein models have been generated and are available over the Internet at http://www.expasy.ch/swissmod/.

In order to provide information about the reliability of the modeling method implemented in 3DCrunch and SWISS-MODEL, we also generated twelve hundred models for proteins of known 3D-structure using modeling templates sharing between 25 and 95 percent sequence identity with the submitted sequences. This is the first time that a modeling method was assessed at this scale, providing relevant information about its reliability. We then derived all degrees of identity between target and template sequences and the relative mean square deviation (base on C^{α} atoms) of the models from their corresponding experimental control structure. First, this allowed us to confirm a few observations:

1. We could confirm the observation (Chothia and Lesk 1986) that the common core of proteins sharing 50 percent sequence identity deviates by approximately 1 Å relative mean square deviation. However, this deviation can be much increased if the protein structures are solved by NMR (Harrison et al. 1995) (also true for structures with identical sequences).
2. We could also confirm the long-known fact that the most reliable part of a protein model is the portion it shares with the modeling template, whereas the rebuilt non-conserved loop were a major contributor to model inaccuracy.

Furthermore, we made three observations that are particularly relevant in the context of automated comparative protein modeling in general and our approach in particular:

1. As shown in table 17.1, 63 percent of the sequences sharing 40–49 percent identity with their template yield a model deviating by less than 3 Å from their control structure. This number increases to 79 percent for sequence identities ranging from 50 to 59 percent. One can also see that below 30 percent sequence identity, the accuracy of completely automated protein modeling is rapidly degrading.
2. Comparative protein modeling is guided by the alignment between target and template sequence. Error introduced by the alignment algorithm will have profound effects on the model. We have observed that these errors start to appear when the sequence identity between target and template sequence is lower than 30–35 percent, which parallels and confirms the rule of thumb set forth by John Moult (Moult 1999). It is noteworthy that less than 5 percent of the models generated by SWISS-MODEL are inaccurate because of such errors. Analyzing several cases, we realized that visual inspection of the alignment and manual optimization of the alignment can in some

Table 17.1
Probabilities of SWISS-MODEL accuracy for target-template identity classes

Percent sequence identity[a]	Total number of models[b]	Percent[c] models with rmsd < 1 Å	Percent models with rmsd < 2 Å	Percent models with rmsd < 3 Å	Percent models with rmsd < 4 Å	Percent models with rmsd < 5 Å	Percent models with rmsd > 5 Å
25–29	125	0	10	30	46	67	33
30–39	222	0	18	45	66	77	23
40–49	156	9	44	63	78	91	9
50–59	155	18	55	79	86	91	9
60–69	145	38	72	85	91	92	8
70–79	137	42	71	82	85	88	12
80–89	173	45	79	86	94	95	5
90–95	88	59	78	83	86	91	9

[a] Range of sequence identity between target and template sequence.
[b] Total number of models in any given class of sequence identity. The table summarizes 1201 model-control structure pairs.
[c] Probability in percent that the Cα atoms of a model, sharing X% sequence identity with its template, deviate by 1 Å or less from the corresponding experimental control structure. The following columns provide these probabilities for other relative mean square deviations.

cases improve the model markedly. This is not a general rule, as there is often no objective function able to distinguish between right and wrong in low similarity segments. Therefore, further benchmarks are needed to assess whether manual or fully automated alignment methods would yield the best results below the 30–35 percent sequence identity threshold (Sternberg et al. 1999).

3. The choice of the "right" modeling template is crucial, as most high deviations between model and experimental control structures can be traced back to the selected modeling templates. Indeed, in table 17.1, we can see, for instance, that 9 percent of the models using templates with sequence identity of 90–95 percent have an rmsd of 5 Å or more. After analysis of a number of those models, we realized that the template and experimental control structures were elucidated under sometimes very different conditions (including the use of NMR versus X-ray crystallography as an experimental method, presence or absence of ligands), which cause major structural differences between highly similar sequences. As set forth later in this chapter, the experimental conditions at the time of structure elucidation strongly impacts the resulting structure. It is thus crucial that the users be aware of the templates that are used by the modeling procedure or make their own informed choice.

Andrej Sali and co-workers at Rockefeller University (Sanchez and Sali 1998) are also addressing very large-scale protein modeling. Using the Saccharomyces cerevisiae genome as a starting point, they have developed a similar automated model-

ing pipeline using PSI-BLAST and MODELLER (Sali and Blundell 1993). Their goal is to build models for as many protein sequences as possible. Although the modeling software cannot be used over the Web, one can easily search their model database (ModBase) (Sanchez et al. 2000) and download the pre-computed models at http://guitar.rockefeller.edu/modbase. Today, their database contains close to seventeen thousand models from 10 complete genomes.

17.5 Web-Based Automated Protein Modeling

In practice, using standard molecular graphics software, comparative protein modeling is a complex activity requiring an expert knowledge of the available methods and software functionality. Furthermore, most freely available software packages do not provide integrated visualization, analysis, and modeling capabilities optimized for protein modeling. Consequently, over the last several years we have concentrated on developing an automated comparative protein modeling server (Peitsch 1995, 1996; Guex and Peitsch 1997) called SWISS-MODEL and its graphical front end Swiss-PdbViewer (Deep View) in order to provide the scientific community with a freely available sequence to structure workbench (Guex et al. 1999). Both server and client application are evolving systems, which are regularly updated to incorporate new algorithms and methods as we develop them. The major goal is to improve the accuracy of the models returned by the server, while providing an ever more flexible and user friendly graphical user interface. The internet site http://www.expasy.ch/swissmod/ provides access to the client software and to the modeling server.

More recently, several new sites have started offering comparative protein modeling capabilities of the Web. A list of these sites can be found at http://www.expasy.ch/swissmod/SM_WEBMODEL.html. It contains links to:

1. CPHmodel (Lund et al. 1997), developed by S. Brunak and coworkers, at the Centre for Biological Sequence Analysis of the Technical University of Denmark. This server can be reached at http://www.cbs.dtu.dk/services/CPHmodels/.

2. SDSC1, developed by I. N. Shindyalov and P. E. Bourne at the San Diego Supercomputer Center. This server can be reached at http://cl.sdsc.edu/hm.html.

3. 3D-JIGSAW (Bates and Sternberg 1999), developed by P. A. Bates and M. J. E. Sternberg at the Imperial Cancer Research Fund. This server can be reached at http://www.bmm.icnet.uk/people/paulb/3dj/form.html.

4. The Full Automatic Modeling System (FAMS) (Ogata and Umeyama 1998), developed by K. Ogata and H. Umeyama at Kitasato University in Japan. This server can be reached at http://physchem.pharm.kitasato-u.ac.jp/FAMS/fams.html.

17.6 Note about Detailed Modeling

As stated above (17.4), it is unclear under which conditions detailed (manual) modeling yields better results than automated approaches. However, it is important to realize that homology modeling can be tricky, especially when the degree of similarity between target and template sequences drops below the 30–35 percent range in some regions of the alignment. It is then generally advisable to further fine-tune the model manually. The resulting models can be of higher quality than their automated counterparts. The process we routinely apply to modeling projects is to build an initial automated model using the first approach mode of SWISS-MODEL. We then fine-tune this model using the interactive modeling features implemented in the Swiss-PdbViewer (Deep View). However, any combination of the above mentioned Web-based modeling servers and detailed modeling packages such as MODELLER (Sali and Blundell 1993) (http://guitar.rockefeller.edu/modeller/modeller.html), COMPOSER (Topham et al. 1990) (http://www-cryst.bioc.cam.ac.uk), CONGEN (Bruccoleri 1993) (http://www.congenomics.com/congen/congen_toc.html), or WHATIF (Vriend 1990) (http://swift.embl-heidelberg.de/whatif) can be used to follow a similar process.

17.7 What Defines the Accuracy of a Model

The quality of a model is determined by two distinct criteria, which will determine its applicability. First, the correctness of a model is dictated by the quality of the sequence alignment used to guide the modeling process. If the sequence alignment is wrong in some regions, then the spatial arrangement of the residues in this portion of the model will be incorrect. The first edition of the community-wide experiment known as critical assessment of protein structure prediction (CASP) already underscored that most severe modeling errors can be traced back to sequence alignment mistakes (Mosiman et al. 1995). This remains, despite many efforts to address this issue (Martin et al. 1997; Jones and Kleywegt 1999), the main weakness of comparative protein modeling. Second, the accuracy of a model is essentially limited by the deviation of the used template structure(s) relative to the experimental control structure. This limitation is inherent to the methods used, as models result from an extrapolation. As a consequence, the core C^{α} atoms of protein models that share 35–50 percent sequence identity with their templates will generally deviate by 1.5 to 1.0 Å from their experimental counterparts, as do experimentally elucidated structures (Chothia and Lesk 1986). However, one should not overlook the contributions of the templates to the model accuracy. The templates, which are obtained through experi-

mental approaches, are subject to structural variations not only caused by experimental errors and differences in data collection conditions, such as the temperature (Tilton et al. 1992), but also because of different crystal lattice contacts and the presence or absence of ligands. Furthermore, X-ray crystallography and NMR generally yield 3D structures with an even broader rmsd spread. This is well illustrated by a typical example: the structure of interleukin-4 (IL-4) (Harrison et al. 1995), a cytokine consisting of a 130-residue four-helix bundle, was elucidated by X-ray crystallography as well as by NMR. The backbones of three IL-4 crystal structures (PDB entries 1RCB, 2INT, and 1HIK) show rms deviations of 0.4 to 0.9 Å, whereas those of three IL-4 NMR forms (PDB entries 1ITM, 1CYL, and 2CYK) deviated by 1.2 to 2.6 Å. These values illustrate the structural differences due to experimental procedures and the molecular environment at the time of data collection. It is thus crucial to know the experimental conditions under which the modeling templates were collected, as this has a direct impact on the accuracy of the derived models and thereby on their potential use.

Almost every protein model contains non-conserved loops, which are expected to be the least reliable portions of a protein model. Indeed, non-conserved loops often deviate markedly from experimentally determined control structures. In many cases, however, these loops also correspond to the most flexible parts of the structure as evidenced by their high crystallographic temperature factors (or multiple solutions in NMR experiments). On the other hand, the core residues—the least variable in any given protein family—are usually found in essentially the same orientation as in experimental control structures, although far larger deviations are observed for surface amino acids. This is expected as the core residues are generally well conserved and the rotamers of their side chains are constrained by neighboring residues. In contrast, the more variable surface amino acids will tend to show more deviations because there are few steric constraints imposed upon them.

Some structural aspects of a protein model can be verified using methods based on the inverse folding approach. Two of them, namely the 3D profile based verification method (Lüthy et al. 1992) and ProsaII, developed by Sippl (Sippl 1993), are widely used. The 3D profile of a protein structure is calculated by adding the probability of occurrence for each residue in its 3D context (Lüthy et al. 1992). Each of the 20 amino acids has a certain probability to be located in one of the 18 environmental classes (defined by criteria such as solvent-accessible surface, buried polar and exposed nonpolar area, and secondary structure) presently defined by Eisenberg and colleagues. In contrast, ProsaII (Sippl 1993) relies on empirical pseudo-conformational energy potentials derived from the pairwise interactions observed in well-defined protein structures. These terms are summed over all residues in a model and result in a

more (more negative) or less (more positive) favorable energy. Both methods can detect a global sequence to structure incompatibility and errors corresponding to topological differences between template and target. They also allow the detection of more localized errors such as β-strands that are "out of register" or buried charged residues. These methods are, however, unable to detect the more subtle structural inconsistencies often localized in non-conserved loops, and cannot provide an assessment of the correctness of their geometry.

Finer details of a model, especially deviations from the ideal stereochemistry (bond length, dihedral angles, etc.), internal residue packing, and so on, can be easily analyzed with programs such as WhatCheck (Hooft et al. 1996) or PROCHECK (Laskowski et al. 1993). It is, however, crucial to realize that proper stereochemistry is not sufficient as a criterion for model correctness or value. It is possible, if not easy, to build models complying with all tests implemented in WhatCheck or PROCHECK but having little or no biological meaning. Nevertheless, these programs allow us to select the best model among a number of very closely related ones, especially when stereochemistry is the last distinguishing feature between a collection of models.

17.8 About the Use of Protein Models

Protein models obtained with comparative modeling methods can be classified into three broad categories: (1) models that are based on incorrect alignments between target and template sequences. Such alignment errors, which generally reside in the inaccurate positioning of insertions and deletions, are caused by the weaknesses of the alignment algorithms and often cannot be resolved in the absence of a control experimental structure. It is, however, often possible to correct such errors by producing several models based on alignment variants and by selecting the most "sensible" solution. Nevertheless, it turns out that such models are often useful as the errors are not located in the area of interest, such as within a well-conserved active site. (2) Models based on correct alignments are of course much better, but their accuracy can still be medium to low as the templates used during the modeling process have a medium to low sequence similarity with the target sequence. Such models, as the ones described above, are still very useful tools for the rational mutagenesis experiment design. They cannot be of great assistance during detailed ligand binding studies. (3) The last category of models comprises all those that were built based on templates that share a high degree of sequence identity (> 70%) with the target. Such models have proven useful during drug design projects and allowed the taking of key decisions in compound optimization and chemical synthesis. For instance, models of

several species variants of a given enzyme can guide the design of more specific non-natural inhibitors.

However, nothing is absolute and there are numerous occasions in which models falling in any of the above categories could either not be used at all or in contrast prove to be more useful and correct than initially thought. In our experience, several applications of medium-accuracy models have proven successful. These can be classified into three categories.

17.8.1 Interpreting the Impact of Mutations on Protein Function: Potential Link to Diseases

One of the first uses one can make of a model structure is to interpret the impact a mutation can have on the overall function of a protein. Although the development of objective scoring functions has begun only recently, "visual inspection" associated with a good knowledge of the rules underlying protein structure has proven useful in defining the broad reasons for mutant malfunction (Hahne et al. 1995; Lalioti et al. 1997; Notarangelo et al. 1996). With the upcoming high-throughput production of single nucleotide polymorphisms (SNP), objective scoring functions will be crucial to make maximum use of the information. Indeed, a sizeable proportion of the SNPs will alter the translated protein sequences, and thus interpreting the potential functional effects of these mutants will be crucial to elucidate the molecular basis of human diseases.

17.8.2 Prioritization of Residues to Mutate to Determine Protein Function

As mentioned previously, the discovery of gene function in the genomic era will require a sustained experimental effort, including the creation of molecular mutants. The prioritization of residues to mutate will be greatly optimized by considering the 3D structure of the target protein (Peitsch and Tschopp 1995; Schneider et al. 1997).

17.8.3 Providing Hints for Protein Function

This is probably the broadest and least defined spectrum of potential applications for 3D models. The common feature of these applications is that models can be used to formulate a hypothesis around a protein, which can then be tested in experimental settings. It is well known that low, yet significant, degrees of sequence similarity are often not sufficient to attribute a function to a protein. In such cases, protein modeling can provide useful insights and help determine or confirm a potential functional assignment (Duret et al. 1998; Peitsch and Jongeneel 1993). Furthermore, one can

use models to create hypotheses around potential enzymatic activities (Peitsch and Boguski 1991) and possible ligand binding functions (Peitsch and Boguski 1990).

17.9 Modeling Membrane Proteins

Membrane proteins still remain a class of proteins that represent an even greater challenge to modelers. G-protein coupled receptors in particular also represent a group of molecules of special interest to the pharmaceutical industry, as a very large proportion of today's medicines are modulators of their activities. Modeling such proteins has thus been attempted in many occasions by both de novo (Donnelly et al. 1993; Sankararamakrishnan and Sansom 1996; Thomas 1996; Herzyk and Hubbard 1998) and comparative approaches (Thomas 1996). The two main steps along the path to a model have been automated: (1) algorithms have been developed to identify the transmembrane domains (Persson et al. 1996; von Heijne 1992) and (2) to generate 3D models using de novo approaches (Donnelly et al. 1993; Herzyk and Hubbard 1995) and comparative methods (Peitsch et al. 1996). In all cases, however, the sequence analysis and coordinate generation steps were separated and could not be linked automatically due to the relatively low reliability of the first step. Consequently, this group of proteins is not yet amenable to high-throughput model building. This will of course dramatically change with the future availability of experimentally determined structure of one of their family members.

17.10 The Future

Over the next years we will focus on two main aspects of comparative protein modeling: (1) improving the sensitivity of the template identification and selection procedure in the sequence similarity "twilight zone," and (2) improving the model accuracy.

17.10.1 Increasing the Template Selection Sensitivity

The availability of iterative local similarity search algorithms (Altschul et al. 1997) allows for the identification of suitable template structures even at very low sequence similarity levels. In cases where the search results are not significant enough to choose a template, multiple models for a family of sequences based on several possible templates can be constructed. Evaluating the multiple model assembly will then enable the discrimination of compatible template folds. This would allow us to choose the correct template even beyond the limit of the search algorithms, provided a reliable

function discriminating "good" from "wrong" models can be defined over a broad range of different types of errors.

17.10.2 Improving the Accuracy of Models

In order to improve both model correctness and accuracy, we will address two of the main causes of modeling errors. First, there is the sequence alignment, which guides the modeling procedure. Automated sequence alignment procedures often introduce errors by placing insertions and deletions ("indels") incorrectly. This phenomenon is mainly observed when the overall sequence identity between target and template sequences drops below 35 percent. The identification and subsequent multiple alignment of all members of a protein family results in more accurate "indel" placement.

Second, side chain and loop rebuilding are also among the main causes of model inaccuracies. The increasing number of available high-resolution X-ray structures will improve the model accuracy by providing more high-quality templates. In addition, this will allow deriving improved secondary structure specific side chain rotamer libraries and broader loop databases. Both will contribute to the improvement of model accuracy.

References

Altschul, S. F., Gish, W., Miller, W., Myers, E. W., and Lipman, D. J. (1990). Basic local alignment search tool. *J. Mol. Biol.* 215: 403–410.

Altschul, S. F., Madden, T. L., Schäffer, A. A., Zhang, J., Zhang, Z., Miller, W., and Lipman, D. J. (1997). Gapped BLAST and PSI-BLAST: A new generation of protein database search programs. *Nucl. Acids Res.* 25: 3389–3402.

Bajorath, J., Stenkamp, R., and Aruffo, A. (1993). Knowledge-based model building of proteins: Concepts and examples. *Protein Sci.* 2: 1798–1810.

Bates, P. A., and Sternberg, M. J. E. (1999). Model building by comparison at CASP3: Using expert knowledge and computer automation. *Proteins* S3: 47–54.

Bernstein, F. C., Koetzle, T. F., Williams, G. J. B., Meyer, E. F., Brice, M. D., Rodgers, J. R., Kennard, O., Shimanouchi, T., and Tasumi, M. (1977). The protein data bank: A computer-based archival file for macromolaecular structures. *J. Mol. Biol.* 112: 535–542.

Brooks, B. R., Bruccoleri, R. E., Olafson, B. D., States, D. J., Swaminathan, S., and Karplus, M. (1983). CHARMM: A program for macromolecular energy, minimization and dynamics calculation. *J. Comp. Chem.* 4: 187–217.

Bruccoleri, R. E. (1993). Application of systematic conformation search to protein modelling. *Molecular Simulations* 10: 151–174.

Chothia, C., and Lesk, A. M. (1986). The relation between the divergence of sequence and structure in proteins. *EMBO J.* 5: 823–826.

Donnelly, D., Overington, J. P., Ruffle, S. V., Nugent, J. H. A., and Blundell, T. L. (1993). Modelling α-helical transmembrane domains: The calculation and use of substitution tables for lipid facing residues. *Protein Sci.* 2: 55–70.

Duret, L., Guex, N., Peitsch, M. C., and Bairoch, A. (1998). New insulin-like protein with atypical disulphide bond pattern characterised *Caenorhaditis elegans* by comparative analysis and homology modelling. *Genome Research* 8: 348–353.

Greer, J. (1991). Comparative modelling of homologous proteins. *Methods Enzymol.* 202: 239–252.

Guex, N., Diemand, A., and Peitsch, M. C. (1999). Protein modelling for all. *TiBS* 24: 364–367.

Guex, N., and Peitsch, M. C. (1997). SWISS-MODEL and the Swiss-PdbViewer: An environment for comparative protein modelling. *Electrophoresis* 18: 2714–2723.

Hahne, M., Peitsch, M. C., Irmler, M., Schröter, M., Lowin, B., Rousseau, R., Bron, C., Renno, T., French, L., and Tschopp, J. (1995). Characterisation of the non-functional Fas ligand of *gld* mice. *Int. Immunol.* 7: 1381–1386. ·

Harrison, R. W., Chatterjee, D., Weber, I. T. (1995). Analysis of six protein structures predicted by comparative modeling techniques. *Proteins: Struct. Func. Genet.* 23: 463–471.

Herzyk, P., and Hubbard, R. E. (1995). Automated method for modeling seven-helix transmembrane receptors from experimental data. *Biophys. J.* 69: 2419–2442.

Herzyk, P., and Hubbard, R. E. (1998). Using experimental information to produce a model of the transmembrane domain of the ion channel phospholamban. *Biophys. J.* 74: 1203–1214.

Hooft, R. W. W., Vriend, G., Sander, C., and Abola, E. E. (1996). Errors in protein structures. *Nature* 381: 272.

Huang, X., and Miller, M. (1991). A time-efficient, linear-space local similarity algorithm. *Adv. Appl. Math.* 12: 337–357.

Jones, T. A., and Kleywegt, G. J. (1999). CASP3 comparative modelling evaluation. *Proteins* S3: 30–46.

Jones, D. T., and Thirup, S. (1986). Using known substructures in protein model building and crystallography. *EMBO J.* 5: 819–822.

Lalioti, M. D., Mirotsou, M., Buresi, C., Peitsch, M. C., Rossier, C., Ouazzani, R., Baldy-Moulinier, M., Bottani, A., Malafosse, A., and Antonarakis, S. E. (1997). Identification of mutations in *Cystatin B*, the gene responsible for Unverricht-Lundborg type of progressive myoclonus epilepsy. (EPM1). *Am. J. Hum. Genet.* 60: 342–351.

Laskowski, R. A., MacArthur, M. W., Moss, D. S., and Thornton, J. M. (1993). PROCHECK: A program to check the stereochemical quality of protein structures. *J. Appl. Cryst.* 26: 283–291.

Lund, O., Frimand, K., Gorodkin, J., Bohr, H., Bohr, J., Hansen, J., and Brunak, S. (1997). Protein distance constraints predicted by neural networks and probability density functions. *Protein Engineering* 10: 1241–1248.

Lüthy, R., Bowie, J. U., and Eisenberg, D. (1992). Assessment of protein models with three-dimensional profiles. *Nature* 356: 83–85.

Martin, A. C. R., MacArthur, M. W., and Thornton, J. M. (1997). Assessment of comparative modelling in CASP2. *Proteins* S1: 14–18.

Mosimann, S., Melshko, R., and James, M. N. G. (1995). A critical assessment of comparative modelling oftertiary structure of proteins. *Proteins* 23: 301–317.

Moult, J. (1999). Predicting protein three-dimensional structure. *Current Opinion in Biotechnology* 10: 583–588.

Notarangelo, L. D., Peitsch, M. C. et al. (1996). CD40Lbase: A database of CD40L gene mutations causing X-linked hyper-IgM syndrome. *Immunology Today* 17: 511–516.

Ogata, K., and Umeyama, H. (1998). A new method of side chain modelling. *Proteins* 31: 355–369.

Pearson, W. R., and Lipman, D. J. (1988). Improved tools for biological sequence comparison. *Proc. Natl. Acad. Sci. USA* 85: 2444–2448.

Peitsch, M. C. (1995). Protein modelling by e-mail. *Biotechnology* 13: 658–660.

Peitsch, M. C. (1996). ProMod and Swiss-Model: Internet-based tools for automated comparative protein modelling. *Biochem. Soc. Trans.* 24: 274–279.

Peitsch, M. C. (1997). Large scale protein modelling and model repository. In *Proceedings of the Fifth International Conference on Intelligent Systems for Molecular Biology*, vol. 5, Gaasterland, T., Karp, P., Karplus, K., Ouzounis, C., Sander, C., and Valencia, A., eds., 234–236. New York: AAAI Press.

Peitsch, M. C., and Boguski, M. S. (1990). Is apolipoprotein D a mammalian bilin-binding protein? *New Biol.* 2: 197–206.

Peitsch, M. C., and Boguski, M. S. (1991). The first enzyme among the lipocalin family. *TIBS* 16: 363.

Peitsch, M. C., Herzyk, P., Wells, T. N. C., and Hubbard, R. E. (1996). Automated modelling of the transmembrane region of G-protein coupled receptor by Swiss-Model. *Receptors and Channels* 4: 161–164.

Peitsch, M. C., and Jongeneel, V. (1993). A 3-dimensional model for the CD40 ligand predicts that it is a compact trimer similar to the tumor necrosis factors. *Int. Immunol.* 5: 233–238.

Peitsch, M. C., and Tschopp, J. (1995). Comparative molecular modelling of the Fas-ligand and other members of the TNF family. *Mol. Immunol.* 32: 761–772.

Peitsch, M. C., Wilkins, M. R., Tonella, L., Sanchez, J.-C., Appel, R. D., and Hochstrasser, D. F. (1997). Large scale protein modelling and integration with the SWISS-PROT and SWISS-2DPAGE databases: The example of *Escherichia coli*. *Electrophoresis* 18: 498–501.

Persson, B., Milpetz, F., and Argos, P. (1996). Prediction of transmembrane segments in proteins using multiple sequence alignments. In *Membrane Protein Models*, Findlay, J. B. C., ed., 1–25, Oxford: BIOS Scientific Publishers Ltd.

Ponder, J. W., and Richards, F. M. (1987). Tertiary templates for proteins. Use of packing criteria in the enumeration of allowed sequences for different structural classes. *J. Mol. Biol.* 193: 775.

Sali, A., and Blundell, T. L. (1993). Comparative protein modelling by satisfaction of spatial restraints. *J. Mol. Biol.* 234: 779–815.

Sánchez, R., Pieper, U., Mirkovi, N., de Bakker, P. I. W., Wittenstein, E., and Sali, A. (2000). MODBASE, a database of annotated comparative protein structure models. *Nucl. Acids Res.* 28: 250–253.

Sanchez, R., and Sali, A. (1998). Large-scale protein structure modeling of the Saccharomyces cerevisiae genome. *Proc. Natl. Acad. Sci. USA* 95: 13597–13602.

Sankararamakrishnan, R., and Sansom, M. S. P. (1996). α-helix bundles and ion channels. In *Membrane Protein Models*, Findlay, J. B. C., ed., 55–72. Oxford: BIOS Scientific Publishers Ltd.

Schneider, P., Bodmer, J. L., Holler, H., Matmann, M., Scuderi, P., Terskikh, A., Peitsch, M. C., and Tschopp, J. (1997). Characterization of the Fas (Apo-1, CD-95)-Fas ligand (Apo-1L, CD95L) interaction. *J. Biol. Chem.* 272: 18827–18833.

Sippl, M. J. (1993). Recognition of errors in three-dimensional structures of proteins. *Proteins: Struct. Funct. Genet.* 17: 355–362.

Sternberg, M. J. E., Bates, P. A., Kelly, L. A., and MacCallum, R. M. (1999). Progress in protein structure prediction of CASP3. *Current Opinion in Structural Biology* 9: 368–373.

Thomas, P. (1996). Making and breaking models of G protein-coupled receptors. In *Membrane Protein Models*, Findlay, J. B. C., ed., 73–89. Oxford: BIOS Scientific Publishers Ltd.

Tilton, R. F., Dewan, J. C., and Petsko, G. A. (1992). Effects of temperature on protein structure and dynamics: X-ray crystallographic studies of the protein ribonuclease-A at nine different temperatures from 98 to 320 K. *Biochemistry* 31: 2469–2481.

Topham, C. M., Thomas, P., Overington, J. P., Johnson, M. S., Eisenmenger, F., and Blundell, T. L. (1990). An assessment of Composer—a rule based approach to modelling protein structure. *Biochemical Society Symposium* 57: 1–9.

Van Gunsteren et al. (1996). In Biomolecular Simulation: The GROMOS96 Manual and User Guide. Vdf Hochschulverlag ETHZ.

Venclovas, C., Zemla, A., Fidelis, K., and Moult, J. (1999). Criteria for evaluating protein structures derived by comparative modelling. *Proteins* S1: 7–13.

Von Heijne, G. (1992). Membrane protein structure prediction. Hydrophobicity analysis and the positive-inside rule. *J. Mol. Biol.* 225: 487–494.

Vriend, G. (1990). WHAT IF: A molecular modeling and drug design program. *J. Mol. Graph.* 8: 52–56.

Westhead, D. R., and Thornton, J. M. (1998). Protein structure prediction. *Current Opinion in Biotechnology* 9: 383–389.

18 Protein Structure Prediction by Protein Threading and Partial Experimental Data

Ying Xu and Dong Xu

18.1 Introduction

The Human Genome Project and other genome sequencing efforts are producing DNA sequence at a prodigious rate, yielding thousands to millions of new genes and proteins. A major challenge facing the entire biological community is to understand the functions of these newly identified proteins in biological systems and how the functions are performed at the molecular level. The three-dimensional (3D or tertiary) structures provide the essential information for defining the proteins' biological functions. In addition, the 3D structures provide a key to engineering proteins and designing drugs targeted at proteins related to diseases. As we move toward the post-genome era, we expect that the demand for rapid protein structure determination will grow drastically. Traditional experimental methods for protein structure determination alone, such as X-ray crystallography and NMR (nuclear magnetic resonance), will probably not be able to keep up with the pace at which protein sequences are being generated. Computational methods could play a significant role, in conjunction with experimental methods, in protein structure determination on a genome-scale.

The earliest work on prediction of protein tertiary structures can be traced back to 1970s (Levitt and Warshel 1975). Significant advancement has been made in the capability of protein structure prediction since then. In the past several years, many nontrivial structure predictions (Nilges and Brünger 1993; Hu et al. 1995; Madej et al. 1995; de las Alas et al. 1998; Villoutreix et al. 1999), prior to the experimental structures, turned out to be fairly accurate. Most notably, the success of protein structure prediction has been demonstrated in the community-wide experiments on the "critical assessment of techniques for protein structure prediction" (CASP) (CASP, 1995, 1997, 1999), which started about six years ago. In each of the past three CASP experiments and the ongoing CASP4, predictors are given a list of protein sequences whose structures have been solved experimentally (or are expected to be solved during the CASP prediction season) but unpublished. The prediction teams submit their predictions before a preset expiration date for each prediction target. Their prediction results are then evaluated against the experimental structures at the end of the prediction season. It has been shown that many predictions in CASP have good enough qualities to make some functional inferences. Even in cases where the entire predicted structure is not accurate enough to make a direct functional analysis, the predictions, when in conjunction with supporting experimental data, can provide valuable insights into the structure and function of a protein. In particular, predicted structures can often give experimenters some leads for further studies.

There are two main classes of tertiary structure prediction methods: ab initio structure prediction, and template-based structure prediction. Ab initio prediction methods (Li and Scheraga 1987; Friedrichs and Wolynes 1989; Skolnick and Kolinski 1991; Sali et al. 1994; Pedersen and Moult 1997) attempt to predict protein structure based on physi-chemical principles directly. A comprehensive review of ab initio methods appears in chapter 16. Template-based methods use known 3D structures in PDB (Protein Data Bank) (Bernstein et al. 1977) as templates to derive the structure of a query protein. Homology modeling and fold recognition methods belong to this class. A *homology modeling* method (also known as *comparative modeling*) (Sali and Blundell 1993; Srinivasan and Blundell 1993; Peitsch 1996) predicts the 3D coordinates of atoms of a query protein mainly based on an optimal sequence alignment between the query protein and a template protein with known structure. It is discussed in detail in chapter 17. *Fold recognition* methods attempt to recognize the "correct" templates from a structure library for the query protein and generate an alignment between the query and the recognized template protein, from which the backbone structure of the query protein can be predicted. Recognition of the correct templates are typically achieved by either the sequence profile approach or the sequence-structure comparison (or threading) approach. *Sequence profile methods* (Krogh et al. 1994; Altschul et al. 1997) detect remote homologs through recognizing similar sequence profiles exhibited by the templates and by the query protein and its homologs through multiple sequence alignments. Threading identifies a homolog or analog through aligning the query protein sequence onto template structures. The main difference between threading and sequence profile methods is that threading uses not only information related to sequence but also structure information such as residue-residue contact patterns in a structure. As researchers start to develop hybrid methods through combining information from different methods, the boundaries between these methods become more and more blurred. For example, one can use the alignment derived from fold recognition in homology modeling, or assemble partial structures predicted by threading in an ab initio prediction.

Each prediction method has its own strength and limitations. In theory, ab initio methods are the most general prediction methods, but at the current stage, they are far from being practical due to their poor prediction reliability and long computing time, though a few isolated successful predictions have been reported. Homology modeling, as a relatively mature method, has been widely used. It can provide relatively reliable atomic coordinates with a low root mean square deviation (rmsd) between a model and a high-quality experimental structure. However, homology modeling applies to only proteins having structure templates with high sequence similarity. Fold recognition, which does not require significant sequence similarity

between a query and the template protein, has gained significant ground in recent years. It has substantially extended the applicability of template-based methods and often generates useful backbone structures. When compared with ab initio methods, the search space for threading is intrinsically discrete rather than continuous, implying a much lower computational cost (time) needed for searching for an optimally folded state. Among these prediction methods, protein threading probably applies to the largest class of proteins, though it is limited to backbone predictions only.

A new trend in structure prediction is to incorporate partial experimental data as constraints in the computation process, blurring the boundary between structure prediction and determination. Such data may include anything that is relatively easy to obtain and helpful for structure calculation, ranging from disulfide bonds, active sites, cross-linking data from mass spectrometry, or sparse NMR data, to X-ray or neutron scattering data. The merge of sophisticated computational methods and partial experimental data could potentially lead to new and more efficient paradigms of protein structure determination.

This chapter focuses on protein structure prediction based on protein threading. We will first introduce the fundamentals of protein threading and use our own program PROSPECT (Xu et al. 1998a; Xu and Xu 2000) as an example to illustrate the basic ideas of threading methods. Then we will describe enhanced threading methods with partial experimental data. In the last part of this chapter, we will discuss challenging issues and future outlook of the threading method. A reader can find more information about threading from the cited literature and the Web pages listed in the appendix.

18.2 Fundamentals of Protein Threading

The idea of threading was originated from the observation that proteins with no apparent sequence similarity could have similar structural folds[1] (Levitt and Chothia 1981; Finkelstein and Ptitsyn 1987). Recent studies have further indicated that the total number of different structural folds in nature may be quite small (Li et al. 1996; Wang 1998), possibly in the range of a few thousand or even fewer, which is at least two orders of magnitude fewer than the number of known protein sequences. Having realized this, a structure prediction problem can be potentially reduced to a recognition problem—which fold(s) the query protein sequence will fold into—plus a modeling problem of side chains. The realization shifted the paradigm of protein

1. A structural fold is the 3D conformation of a protein's backbone.

structure prediction. An inverse folding problem (Drexler 1981; Pabo 1983; Ponder and Richards 1987) was then formulated—searching for sequences that are compatible with a specified structural fold based on sequence-structure relationships. It has long been known that different amino acids may prefer different structural environments: for example, a hydrophobic amino acid tends to be in the interior of a globular protein, and proline rarely occurs in an α-helix. In addition, certain amino-acid pairs (e.g., salt bridges) in space are more favorable than other pairs. Using these sequence-structure relationships, one can differentiate which sequence-structure pairs are more favorable in order to predict which structure is most likely to be adopted by a given sequence. This new paradigm for protein structure prediction, called "threading," achieved its initial success by the pioneering work of Eisenberg and colleagues (Bowie et al. 1991; Luthy et al. 1992). Since then, threading has been developed into a widely used technology for structure prediction through the further efforts of many other research groups (Sippl and Weitckus 1992; Jones et al. 1992; Godzik et al. 1992; Bryant and Altschul 1995; Fischer et al. 1996b; Alexandrov et al. 1996; Xu et al. 1998a).

The computational formulation of threading can be summarized as follows. Given a query protein sequence **s** of unknown structure, threading searches a structure template library **T** to find the best sequence-structure pair **s-t**, $\mathbf{t} \in \mathbf{T}$, measured by the overall preference of individual residues to their structural environment and of residue-residue contacts. A threading method typically consists of four components (Smith et al. 1997): (1) a library **T** of representative 3D protein structures for use as templates; (2) an energy function for measuring the fitness between a query **s** and a template **t**, where $\mathbf{t} \in \mathbf{T}$; (3) a threading algorithm for searching for the lowest energy among the possible alignments for a given **s-t** pair; and (4) a criterion for estimating the confidence level of the predicted structure. In the rest of this section, we will address each aspect in detail.

18.2.1 Fold Templates

Protein threading rests on the premise that the 3D structures of proteins have been better conserved during evolution than their sequences. Proteins can be classified into groups, at different levels, according to their structural and evolutionary relationships. A widely used classification scheme consists of three levels of groups: family, superfamily, and fold (Murzin et al. 1995). A *family* consists of proteins that have high sequence identity among each other and share a common evolutionary ancestor. Proteins of different families sharing a common evolutionary origin (reflected by their common structural and functional features) are placed in the same *superfamily*. Different superfamilies are grouped into a *fold* family if their proteins have the same

major secondary structures in the same arrangement and with the same topological connections. The structural similarities among proteins of the same fold family (but not the same superfamily) may arise just from the protein energetics favoring certain packing arrangements instead of a common evolutionary origin. The structural difference generally gets larger among proteins of the same family, superfamily, and fold.

Most template libraries of the existing threading programs are based on three widely used databases of protein structure classifications: *CATH* (Orengo et al. 1997), *FSSP* (Holm and Sander 1996), and *SCOP* (Murzin et al. 1995). *CATH* is a hierarchical classification of protein domain structures. *FSSP* contains the similar information but is based on protein chains rather than domains; in addition, it contains sequence neighbors and multiple structure alignments. *SCOP* uses a manual procedure to classify protein domains into folds, superfamilies, and families. Hence, its classification is probably of higher quality than the other two. However, *SCOP* has not been updated as frequently as desired simply due to the amount of manual work involved, whereas *FSSP* and *CATH* have been following the Protein Date Bank (PDB) updates closely. The classifications for folds by the three databases differ somewhat due to their different classification criteria (e.g., classification on a whole chain or a structure domain) and structure-structure comparison methods. As a result, the number of folds (or *unique* folds) differ among the three databases. Based on the data available as of June 2000, CATH contains 580 folds (updated June 2000), FSSP has 556 folds (updated June 2000), and SCOP consists of 548 folds (updated February 2000).

In principle, one can use one representative from each fold family as the template library for threading. However, it is often helpful to include a representative for each superfamily or even each family in the template library to cover variations among structures of a fold family and to achieve better prediction accuracy. In designing the template library, one can use both protein chains as well as protein domains.

The general applicability of protein threading is based on the widely held belief that there are only a small number of protein folds. The argument used to support this is that the existing protein structures have evolved from a small number of "primordial folds" (Dorit et al. 1990). During evolution, although protein sequences have diverged significantly, their 3D structural folds have been more conserved and maintained a small number. Another line of argument for the small size of the fold universe is a stereo-chemical one—the vast majority of geometrically possible folds are simply not energetically favorable enough to exist due to stereo-chemical constraints and physical interactions (Finkelstein and Ptitsyn 1987). This suggestion is supported by a computer simulation of protein folding using a simple lattice model (Li et al. 1996). The simulation has demonstrated that only a handful of proteinlike

folds can adopt many different sequences and keep the energy favorable, whereas other folds do not. The results suggest that protein structures have been selected in nature because they are readily designed and stable against mutations, and that such a selection simultaneously leads to thermodynamic stability. This line of argument seems to be convincing and may possibly explain the assumption used in the first argument, that the number of primordial folds was small.

The next question is how many unique folds for globular proteins exist in nature. Many suggest that this number is less than a thousand, so about half of all possible unique folds are already known (Chothia 1992; Wang 1996; Zhang and DeLisi 1998). A recent estimate (Wang 1998) even suggests that the total number of unique folds may be as low as 650. These estimates are based on statistical analysis of the novel-fold occurrence in the PDB. However, there is a danger in deriving incorrect conclusions from such an analysis due to the possibility that the distribution of the PDB structures is biased. For example, some types of proteins are difficult to purify or are not soluble enough to have either crystal structures or NMR structures. It is quite possible that one thousand or fewer folds may be an underestimate. On the other hand, some have suggested that the number of unique folds could be as high as eight thousand (Orengo et al. 1994). The argument was that 3 percent of sequence databases has yielded 80 folds at the time of estimation. If the sequences represent one-third of all superfamilies, the number of folds should be $80 \times 3/0.03 = 8{,}000$. The underlying assumption used here is that every fold is equally populated in protein sequences. This is evidently incorrect. For example, the TM barrel fold occurs much more frequently than other folds. Hence, the eight thousand folds may be an overestimate. Though it is still an open question about the number of folds in nature, it is clear that the percentage of proteins being submitted to PDB that have new folds has been decreasing. As one can see from the statistics at the PDB Web site, http://www.rcsb.org/pdb/holdings.html, this percentage has dropped from about 30 percent in the 1980s to about 10 percent during the past three years. These numbers may have a more practical meaning for threading: given a new sequence of globular protein, what is the probability that it has a template in PDB? The Live Bench program (http://BioInfo.PL/LiveBench/1/) recently carried out a survey on 125 structures submitted to PDB between October 29, 1999, and April 6, 2000. Only five proteins had no structurally similar proteins in PDB; 96 percent proteins submitted to PDB during this period are not unique folds. This percentage may be a bit too high for globular proteins genome-wide, given the biased distribution of proteins in PDB. However, it is safe to say that the percentage should be at least 50–70 percent (Jones 1999). This number is clearly growing as more and more structures are being solved and put into PDB.

18.2.2 Energy Function

The energy function describes how favorable an alignment between a query sequence and a template structure is. Threading generally uses knowledge-based energy functions rather than physical energies. The main consideration is that even though a query protein may share the same fold of the template, the exact 3D coordinates of their corresponding atoms may differ to a degree that the physical energy (including bond, van der Waals, electrostatics based on CHARMM potential [Brooks et al. 1983]) of the correct query-template alignment may not be favorable. This is simply because these physical energies are too sensitive to small displacement of atomic coordinates, making them less suitable for threading. Another consideration comes from the fact that the calculation of these physical energies is too time-consuming. A recent study has shown that knowledge-based threading energy by and large captures the detailed atomic potentials (Mohanty et al. 1999).

Theoretically, one could build a probabilistic model of knowledge-based threading energy that accounts for different factors and the correlations between these factors. Such a model could directly give the probability of adopting a certain fold by a given sequence. However, this type of model has not been successfully developed yet, due to our limited understanding of the complicated factors that determine a fold. Currently, most researchers use simplified energy functions without calculating probabilities or considering any correlation between different energy terms. A typical threading energy function has the following form:

$$E_{total} = E_{mutate} + E_{gap} + E_{single} + E_{pair} \tag{18.1}$$

The mutation energy E_{mutate} describes the compatibility of substituting one amino acid type by another; E_{gap} is the alignment gap penalty; the singleton energy E_{single} represents a residue's preference to its local secondary structures (α-helix, β-strand, and loop) and its preference to being in certain solvent environment (either exposed to solvent or in the interior of the protein); E_{pair} is the pairwise potential between spatially close residues that are not neighbors in the protein sequence. The different energy terms are weighted, as elaborated later in section 18.32.

The mutation energy and the alignment gap penalty are similar to the ones used in sequence alignments. The original concept of threading did not include the mutation energy but assumed that sequence-structure alignment (i.e., E_{single} and E_{pair}) should be sufficient for fold recognition. Prediction practices in the past few years have clearly demonstrated the usefulness of the mutation energy, so, most existing threading programs include it. Several matrices have been developed based on mutation rates found in sequence databases; the most popular ones are the PAM (Dayhoff 1978)

and BLOSUM (Henikoff and Henikoff 1992) matrices. BLOSUM-62 is a widely used matrix for searching for close homologs, whereas PAM250 (Gonnet et al. 1992) is more suitable for identifying remote homologs. It has been shown that PAM250 is one of the best substitution matrices available for threading (Fischer et al. 1996a; Abagyan and Batalov 1997). The gap penalty is often a linear function of the gap size, with a penalty for opening a gap and a small penalty for each extension thereafter.

Both E_{single} and E_{pair} are typically derived from Boltzmann statistics from a non-homologous protein database. The basic idea is that if an amino acid is frequently observed in the interior of protein structures, a favorable energy value will be rewarded when it is aligned to an interior position of a template. Though most of the threading programs follow a similar principle in implementing their energy functions, detailed variations may exist. For example, some use the degree of environmental polarity (the degree of burial by polar rather than apolar atoms [Bowie et al. 1991]) and/or secondary structure fitness between the template and the query protein (Rost 1995), whereas others may not explicitly use the singleton term (Bryant and Lawrence 1993) as they believe that the information of singleton energy is covered in the pairwise term (e.g., a favorable packing between hydrophobic residues reflects a similar information of solvent accessibilities). The E_{pair} term is based on a pairwise energy parameter $e_{pair}(i, j)$, which describes the preference for the combination of two spatially close residues of type i and j (say, up to 15 Å between the C_β atoms). Some threading programs further divide this range into subintervals and have a different preference value for each subinterval (Sippl and Weitckus 1992; Jones et al. 1992; Bryant and Lawrence 1993), whereas others simply treat the whole range as one interval (Alexandrov et al. 1996; Xu et al. 1998a). Different programs may measure the distance between two residues somewhat differently. Some measure the distance between the C_β atoms (Sippl and Weitckus 1992; Alexandrov et al. 1996), whereas others may measure between the backbone atoms (Jones et al. 1992), or even between projected centroids of side chains (Bryant and Lawrence 1993). There are also more sophisticated methods for measuring residue-residue contacts, such as "visible volume" (Lo Conte and Smith 1997) or Voronoi contacts (Zimmer et al. 1998).

We now use our own threading program PROSPECT (Xu and Xu 2000) as an example to illustrate how the E_{single} and E_{pair} terms are calculated. We calculate E_{single} based on a singleton energy parameter $e_{single}(i, ss, sol)$ (Xu et al. 2000a), that is, $E_{single} = \sum_i e_{single}(i, ss_i, sol_i)$. $e_{single}(i, ss, sol)$ describes the energy or preference for a particular combination of amino acid type i, secondary structure type ss, and solvent accessibility type sol, that is,

$$e_{single}(i, ss, sol) = -\log \frac{N(i, ss, sol)}{N_E(i, ss, sol)} \qquad (18.2)$$

where log denotes the natural logarithm; $N(i, ss, sol)$ is the number of amino acids of type i in ss with sol, counted from the database; $N_E(i, ss, sol)$ is the average of the reference state—the estimated number of amino acids of type i in ss and sol assuming i, ss, and sol are independent. $N_E(i, ss, sol)$ is calculated by

$$N_E(i, ss, sol) = \frac{N(i)N(ss)N(sol)}{N^2} \tag{18.3}$$

where $N(i)$ is the number of amino acids of type i, $N(ss)$ is the number of residues in secondary structures of type ss, $N(sol)$ is the number of residues with solvent accessibility type sol, and N is the total number of residues (in our database).

Similarly, E_{pair} is calculated based on a pairwise energy parameter $e_{pair}(i, j)$, that is, $E_{pair} = \sum_{i \le j} e_{pair}(i, j)$. In PROSPECT, the cutoff distance for $e_{pair}(i, j)$ is 7.0 Å between C_β atoms, which accounts for most of the important inter-residue interactions (Jones et al. 1992; Alexandrov et al. 1996). We only consider the residue pairs that are separated by at least three amino acids in the protein sequence[2]. $e_{pair}(i, j)$ is derived from the frequency of the inter-residue pairs, that is,

$$e_{pair}(i, j) = -\log \frac{M(i, j)}{M_E(i, j)}, \quad M_E(i, j) = \frac{M(i)M(j)}{M} \tag{18.4}$$

where $M(i, j)$ is the number of pairs between residues of types i and j in the database; $M_E(i, j)$ is the average of the reference state, the estimated number of i-j pairs assuming that residues of types i and j form pairs without any preference. $M(k) = \sum_s M(k, s)$ $(k = i, j)$. M is the total number of pairs—$M = \sum_{ij} M(i, j)$. In the calculation, the number of pairs between residues of types i and j are partitioned equally in $M(i, j)$ and $M(j, i)$ such that $M(i, j) = M(j, i)$. All statistics are collected from our protein database.

18.2.3 Alignment Algorithm

If we do not consider the pairwise energy, a threading problem is essentially the same as a sequence alignment problem. Such a problem can be solved efficiently by a dynamic programming approach (Needleman and Wunsch 1970; Smith and Waterman 1981). There are a number of computer programs that use dynamic programming for their threading problem, such as 123D (Alexandrov et al. 1996), TOPITS (Rost 1995), SAS (Milburn et al. 1998), and the UCLA-DOE Structure Prediction Server (Fischer and Eisenberg 1996). An advantage of a threading algorithm without con-

2. The pairs separated by one or two amino acids in the protein sequence represent local interactions, which are less important in determining an overall fold.

sidering pairwise energy is its speed. However, without pairwise interactions, the threading accuracy is severely compromised (Xu and Xu 2000).

Threading with pairwise terms and variable-length gaps is generally considered to be a very difficult problem. Under the assumption that pairwise interactions between every pair of amino acids in a protein structure should be considered in threading, the problem was proved to be NP-hard (meaning that the problem's intrinsic computational complexity is too high for an efficient rigorous solution) (Lathrop 1994). Two previous existing threading programs with rigorous solutions both have exponential computational complexity (Bryant and Lawrence 1993; Lathrop and Smith 1996).

To overcome the computational difficulty, several approaches have been proposed. One approach is to employ the "frozen approximation," that is, when calculating a pairwise contact potential for a residue it is assumed that its contacting residues are the same in the template and in the query structure. Such an assumption has reduced the general threading problem to a problem solvable by dynamic programming but at the expense of threading accuracy (Zhang et al. 1997; Xu et al. 1998a). Another approach attacks the problem through statistical sampling (Sippl and Weitckus 1992; Jones et al. 1992; Bryant and Altschul 1995; Crawford 1999). Generally this type of approach does not guarantee to find the globally optimal threading alignment,[3] though they typically work better than the frozen approximation approach. We have developed a threading algorithm (Xu et al. 1998a) that guarantees to solve the globally optimal threading problem efficiently under the assumption that residue-residue contact potentials need to be considered only between spatially close pairs in threading. The detailed algorithm will be discussed in the following section.

18.2.4 Assessment of Threading Results

A threading score between a query sequence and a template structure may not provide enough information about whether the template is the "correct" fold. This is because the scores are generally not normalized to the same scale. Consider a query protein and a false template. It is conceivable that their threading score would be improved if we artificially increased the length of the template sequence by appending some irrelevant protein sequence to it without actually changing the "significance" of the threading alignment. So from the threading scores between a query and a pool of templates, we generally cannot tell if the query's correct fold template is in the pool, nor can we always tell which is the correct fold even if it is there.

There have been a number of attempts to "normalize" threading scores so that they can be compared with each other. An early attempt was to use *z-score* (Flockner

3. Note that the threading problem is a discrete and finite problem; hence, a globally optimal solution is always obtainable. The problem is if the global optimum can be found efficiently.

et al. 1995). For the energy E resulting from a particular alignment, the z-score of E is defined as

$$z = \frac{E - \bar{E}}{\sigma} \tag{18.5}$$

where $\bar{E}$ and σ are, respectively, the average and the standard deviation of the energy distribution resulting from the same alignment after re-shuffling the amino acids of the query sequence. However, it has been shown that the z-score is clearly not effective (Marchler-Bauer and Bryant 1997).

There have also been attempts to use the P-value scheme (Karlin et al. 1990; Karlin and Altschul 1990) as a way to assign a meaning to a threading score. P-value, which estimates the probability of having a particular alignment score between two random sequences, have been successfully applied to sequence alignment, thanks to Karlin's seminal work on a rigorous model of gapless alignments (Karlin et al. 1990; Karlin and Altschul 1990). Due to the lack of a rigorous model for threading, the P-values are typically estimated through compiling a "large" number of threading scores between a query sequence and a template after randomly shuffling its residues (Bryant and Altschul 1995). Although some usefulness of the estimated P-value has been demonstrated, the problem of developing a rigorous P-value scheme for threading remains a challenging open problem. The recent work of Jones (1999) seems to have provided a practical way to "normalize" the threading scores. The idea is to feed a neural network the threading scores along with various normalization factors, such as sequence length, and to let the neural network learn to "optimally" combine these factors based on a training set. The result of the training is a mapping from a threading score to a real number between 0 and 1, with an estimated probability that this number corresponds to a correct fold recognition. We will further explain the idea in the next section using our own implementation of this strategy.

18.3 PROSPECT: A Threading-Based Protein Structure Prediction System

PROSPECT (PROtein Structure Prediction and Evaluation Computer Toolkit) (Xu and Xu 2000) is a computer program that we have developed for protein structure prediction. The core of the program is a threading algorithm (Xu et al. 1998a, b), which guarantees to find the globally optimal alignment between a query sequence and a template structure for an energy function as defined in section 18.2.2. Compared with other threading programs, two unique features of the algorithm are (1) that it finds the globally optimal alignment efficiently when using a widely accepted cutoff distance

between residue-residue contacts, 7 Å between the C_β atoms; and (2) that it allows a user to incorporate experimental data as constraints in the threading process, and guarantees to find the globally optimal threading under the specified constraints. Currently it allows following types of constraints: (1) disulfide bonds between specified residues; (2) active sites involving a specified set of residues; (3) long-range NOE (nuclear Overhauser effects) restraints from NMR experiments; (4) secondary structures predicted by computer programs like PHD (Rost and Sander 1993) or determined from chemical shift data of NMR experiments; and (5) position-dependent profiles based on multiple sequence alignment using SAM (Hughey and Krogh 1996). PROSPECT provides a confidence value for each of its predictions being a true fold-recognition. Recently, PROSPECT has been ported to a 184-node IBM/SP3 supercomputer at Oak Ridge National Laboratory. In this section, we will first give an informal description of the threading algorithm and explain how its computational efficiency is achieved. Then we will briefly explain how we assign a confidence value to each threading result, followed by PROSPECT's prediction performance. This section concludes with a short description on the application side of the PROSPECT program.

18.3.1 Algorithm

The basic problem solved by PROSPECT's threading algorithm is to find an alignment between a query sequence and a template structure, which optimizes an energy function as defined in section 18.2.2. To simplify the problem, we have applied two widely adopted assumptions (Bryant and Lawrence 1993; Lathrop and Smith 1996): (1) no gap can occur within a core secondary structure; (2) any pairwise interaction involving a non-core secondary structure is ignored. Figure 18.1 shows a schematic of such an alignment. What makes the threading problem difficult is the consideration of pairwise contacts, which separate the problem from sequence alignment. Consider the example of figure 18.1. When we are aligning the first part of the query sequence with the first core (and its loop regions), we do not have any knowledge about which residues of the query protein will be aligned to the third, fourth, and fifth cores of the template (in the final optimal alignment), and so do not have enough information for calculation of the corresponding pairwise contact potentials. This illustrates why the dynamic programming scheme, which has been widely used for sequence alignment, does not work for the threading problem.

The very basic idea of our solution to the threading problem follows the idea outlined above—considering all possible amino acid types when considering which type of amino acid is aligned to the template position at the other end (e.g., the third, fourth, and fifth cores of the above example) of each arc, and calculating the pairwise

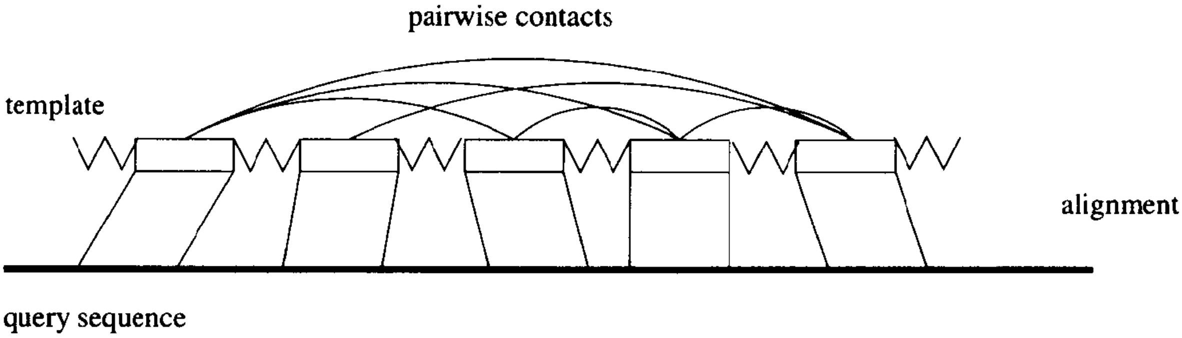


Figure 18.1
A schematic sequence-structure alignment. Each box represents a core secondary structure; the connecting line between two boxes represents a loop region of the template structure. A core secondary structure, or a core, means an α-helix or a β-strand. An arc between two cores indicates that at least one pair of residues from the two cores have contacts (i.e., their C_β atoms are within our cutoff distance). The thick horizontal line represents a query sequence. Lines connecting the query sequence and the template represent a possible sequence-structure alignment.

contact potential for each amino acid type. Clearly doing this directly is not computationally practical because of the enormous number of combinations we need to consider. We have developed an efficient way to implement this idea, which we now explain. Detailed formal description can be found elsewhere (Xu et al. 1998a, b).

The algorithm employs a *divide and conquer* strategy to solve the optimal threading problem. For this purpose, we first pre-process the template by repeatedly dividing (bi-partitioning) it into substructures until each substructure contains only one core secondary structure. Dividing the template cuts an interaction between two cores into two *open links*, represented as an arc with one of its ends being a hollow circle as shown in figure 18.2. Our divide and conquer algorithm works correctly on any bi-partition of the template. However, the way a template is partitioned affects the computing time.

The algorithm solves the entire optimal alignment problem by recursively solving a series of subalignment problems between substructures and subsequences, under various constraints, and then combining these subalignments in a consistent and optimal way. Figure 18.3 illustrates the basic idea, using an example from the last partition step in figure 18.2. In this example, the substructure AB is partitioned into two cores, A and B. The interaction link between A and B in the partition is cut into two open links, that is, a_3 and b_1.

The algorithm first calculates the alignment score between A (similarly B) and each sequence position (meaning that the leftmost residue of A is aligned with that position). Because we assume that there is no alignment gap within a core alignment, this score can be calculated by simply adding the singleton scores, E_{single}, of the aligned

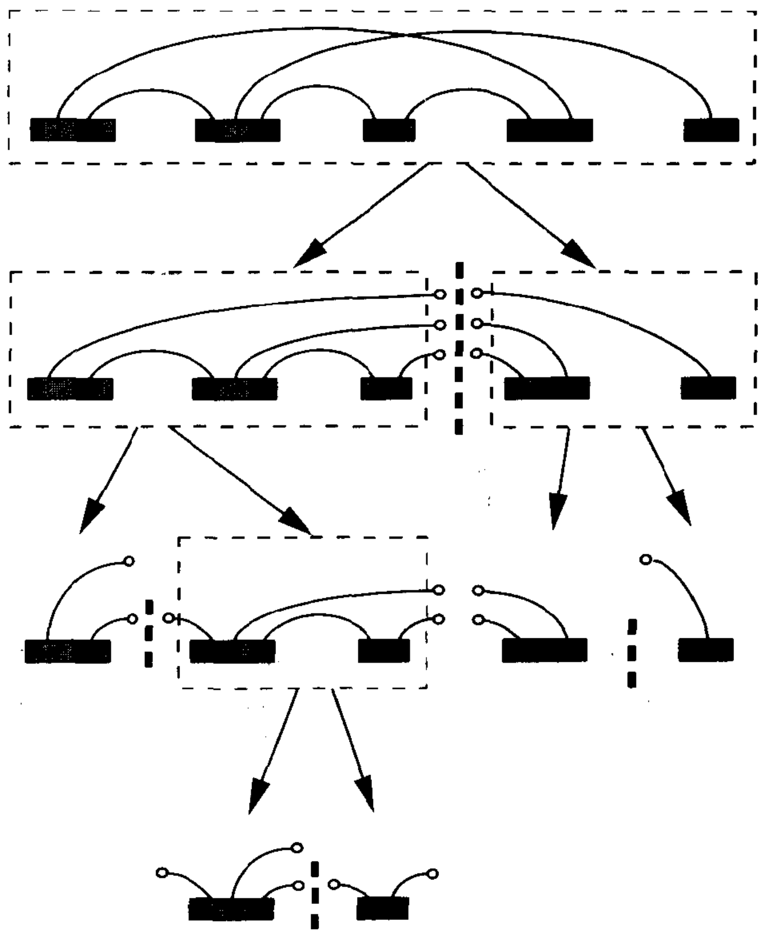


Figure 18.2
A partition for a template structure. The first row shows the template with five core secondary structures. The second row shows a partition of the template into two substructures, one with three cores and the other with two cores. A broken arc ended with a circle is called an *open link*. The third and fourth rows show further partition of the template until each substructure contains only one core secondary structure. Note that a partition forms a tree structure as indicated by the arrows.

residues and structural positions, plus the E_{pair} scores. The calculation of E_{pair} is tricky because we do not know the sequence positions that the cores at the other ends of the open links a_1, a_2, and a_3 (i.e., the first, fourth, and third cores in figure 18.2) are aligned to. Once the other end of an open link is determined, all the pairwise interactions between the two cores connected by the open link can be subsequently calculated. To overcome the missing information, we simply consider all possible *legal* alignments of these cores. Note that not every combination of the alignments of these cores makes a legal (overall) alignment as some of them may (1) violate the relative order of these cores (e.g., the first core is aligned to a sequence position that is to the

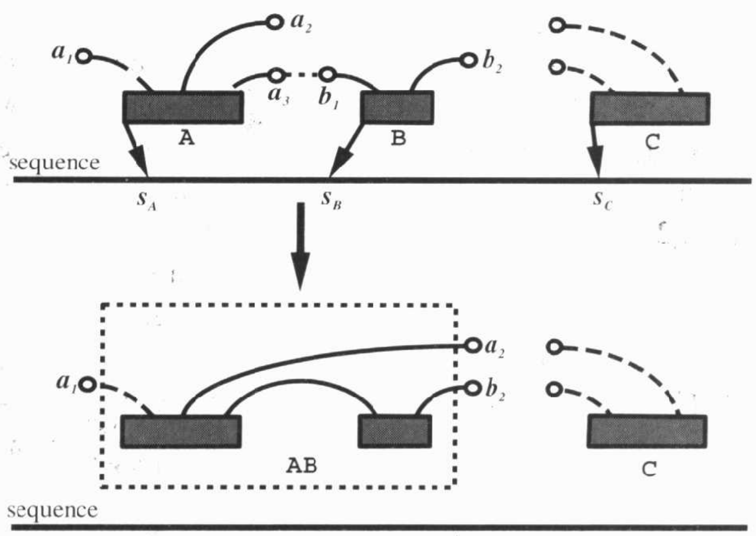


Figure 18.3
A schematic example of the divide-and-conquer algorithm.

right of the aligned sequence position of the fourth core); (2) overlap each other; and (3) violate the allowed minimum and maximum length difference in a loop alignment (we allow a user to specify these numbers in PROSPECT). Though now we have to consider many possible aligned positions of the cores connected with a_1, a_2, a_3 (from now on, we simply call them *assignments* to a_1, a_2, a_3), we have enough information to calculate the pairwise contact term E_{pair} for each fixed assignment to the open links.

To avoid double counting, we treat the two open links created from each interaction differently. We use a dashed arc to represent one open link and a solid arc to represent the other. We only calculate the pair contact energy E_{pair} for the solid arc; both open links are assigned to the aligned position of the core with a dashed arc. Here only a_2 is used to calculate E_{pair}. The corresponding E_{pair} terms for a_1 and a_3 will be calculated when the alignment for the first and the third cores are calculated, respectively. In addition, we define $a_3 = b_1 = s_A$, where s_A is the aligned position of the leftmost residue of core A on the sequence.

After the algorithm calculates the core alignment scores for each core under various assumptions that their open links are assigned to particular sequence positions, it calculates the alignment scores for larger substructures consisting of multiple cores. We continue to use the above example to illustrate the basic idea. We now want to calculate the optimal alignment score for AB under the assumption that a_1, a_2, and b_2 are assigned to particular sequence positions. Calculation of this optimal score consists of two parts: (1) the sum of the alignment scores for A and for B under the

condition that the two partial alignments for A and B are consistent, and (2) the alignment score of the loop connecting A and B. The optimal alignment score for AB is the lowest combined score of (1) and (2) among all possible legal assignments to a_3 and b_1. To find the lowest combined score, the algorithm goes through all the alignment scores for A and B under the conditions (1) that a_3 and b_1 have the same sequence-position assignment, and (2) that $a_2 = b_2$, because they point to the same core (i.e., core C, see figure 18.3). For each such assignment, the algorithm calculates the optimal loop alignment (now the aligned positions of both A and B are fixed) using the dynamic programming method (Smith and Waterman 1981), and adds up the total alignment score for AB. The assignment to a_3 (b_1) that has the lowest combined score gives the optimal alignment score for AB under the specified assignment condition.

This process continues for larger substructures (e.g., the next step is substructure ABC) until the top level of the partition tree is reached, and the whole template is considered. Note that in the lower level calculations, the algorithm repeatedly solves the *constrained* alignment problem, such as finding the optimal alignment score for a substructure under the condition that its open links have particular assignments. On the top level, the whole structure has no open links, and the optimal alignment obtained gives the final solution to our threading problem. We have shown that the final solution of this procedure gives the globally optimal threading for the given energy function defined in section 18.2.2 (Xu et al. 1998a, b).

We have found that the computational bottleneck of this algorithm is the consideration of all legal combinations of link assignments. More specifically, the dominating term in the algorithm's computational complexity (running time) increases exponentially with the maximum number of links among all division points. Fortunately, this (maximum) number is generally a small number if we only consider pairwise contacts between residues that are spatially close, given the optimal way to divide a template. We have previously demonstrated that the maximum number of links changes according to how a template structure is divided (e.g., where to cut first, and then second, etc.), and presented an algorithm for finding a division scheme that minimizes the maximum link number among all division points (Xu et al. 1998a, b). The minimized maximum link number is called the *topological complexity* of a template. The topological complexity is a small number (≤ 8) in all the cases that we have studied when the cutoff distance between the C_β atoms is no more than 15 Å. Table 18.1 shows the distribution of the topological complexity versus the cutoff distance between C_β atoms for pairwise contacts. The statistics were compiled from 750 structures in a subset of the FSSP database (Holm and Sander 1996) (March

Table 18.1
Distribution of topological complexity

	Cutoff (Å)										
TC	5	6	7	8	9	10	11	12	13	14	15
0	52 (7%)	52 (7%)	52 (7%)	52 (7%)	52 (7%)	52 (7%)	52 (7%)	52 (7%)	52 (7%)	52 (7%)	52 (7%)
1	87 (12%)	87 (12%)	87 (12%)	87 (12%)	87 (12%)	87 (12%)	87 (12%)	87 (12%)	87 (12%)	87 (12%)	87 (12%)
2	218 (29%)	162 (22%)	135 (18%)	118 (16%)	112 (15%)	106 (14%)	102 (14%)	98 (13%)	96 (13%)	94 (13%)	92 (12%)
3	364 (49%)	318 (42%)	245 (33%)	189 (25%)	152 (20%)	131 (17%)	114 (15%)	104 (14%)	93 (12%)	83 (11%)	77 (10%)
4	29 (4%)	126 (17%)	216 (29%)	245 (33%)	218 (29%)	198 (26%)	185 (25%)	175 (23%)	161 (21%)	156 (21%)	154 (21%)
5	0 (0%)	5 (.7%)	15 (2%)	58 (8%)	121 (16%)	156 (21%)	162 (22%)	151 (20%)	136 (18%)	120 (16%)	105 (14%)
6	0 (0%)	0 (0%)	0 (0%)	1 (.1%)	8 (1%)	19 (3%)	47 (6%)	82 (11%)	117 (16%)	135 (18%)	141 (19%)
7	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	1 (.1%)	1 (.1%)	1 (.1%)	8 (1%)	23 (3%)	38 (5%)
8	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	4 (.5%)

Each column represents a cutoff distance, and each row represents the number of proteins having a particular topological complexity (TC).

1998 release). The lengths of these proteins range from 31 to 793; the numbers of their core secondary structures range from 1 to 34.

By using a similar analysis to that of Xu et al. (1998a, b), it can be proved that this threading algorithm runs in a time proportional to $mn + Mn^{TC}N^{TC/2}$, where m and n represent the lengths of the template and query protein sequences, respectively; M is the number of core secondary structures of the template; N is the maximum allowed difference between the lengths of two aligned loops; and TC is the topological complexity of the template. This function shows how various parameters affect the running time of the algorithm. Using a recently improved version of PROSPECT, the actual threading time on a workstation is typically less than a minute.

18.3.2 Performance by PROSPECT

In a recent study (Xu and Xu 2000), we tested PROSPECT on 312 pairs of query-template proteins, each pair being from the same superfamily and having a sequence identity $\leq 30\%$. We randomly selected 175 pairs as the training set to obtain the

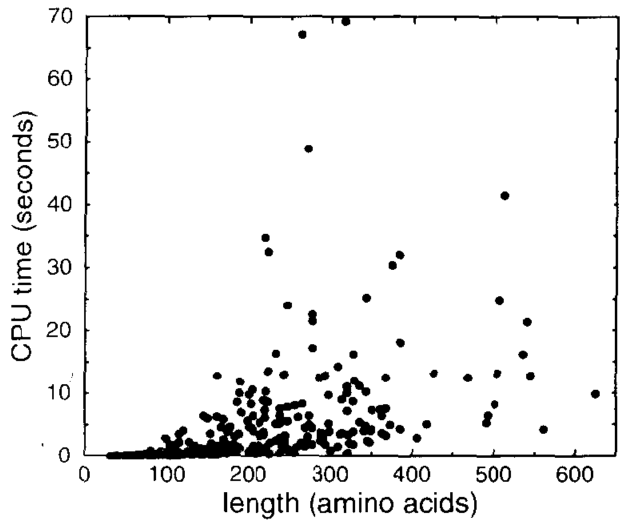


Figure 18.4
CPU time of PROSPECT vs. the length of query sequence for threading each of 312 proteins on the template of its native folds using a Linux workstation with Pentium III 550 MHz. To show the majority of the data clearly, two extreme data points (421 amino acids, 117.9 seconds) and (907 amino acids, 8.4 seconds) are not included within the scale of the figure.

weights between different energy terms, and another 137 pairs as the independent test set to verify the threading performance. Figure 18.4 gives a benchmark of CPU time for threading the 312 pairs on a Linux workstation with a Pentium III 550 MHz CPU.

For each pair of proteins, we put one into the query set and the other into the template set (both proteins of each pair have solved structures). In the fold recognition test, we ran each protein from the query set against the whole template set. Table 18.2 shows the performance of threading accuracies. Overall, PROSPECT recognizes 69 percent of the templates correctly (among top five in ranking) and aligns 66 percent of the structurally alignable residues correctly (defined as being aligned to within a four-residue shift from the SARF's [Alexandrov 1996] structure-structure alignment position). Interestingly, these numbers may be compared with the 55 percent fold recognition and 64 percent alignment accuracy for the same test set without using the pairwise energy, indicating the significant contribution from the pairwise term, particularly for fold recognition. The fold recognition and alignment accuracy are further improved to 72 percent and 74 percent, respectively, when the secondary structure information predicted by the PHD program is used in scoring.

18.3.3 Normalization of Threading Scores

The goal of normalizing PROSPECT's threading scores is to map all its scores to a fixed range, say [0, 1], so that we can assign a fixed meaning to each normalized

Table 18.2
Threading performance vs. sequence identity

seq identity		1–6%	7–9%	9–12%	13–15%	16–18%	19–21%	22–24%	25–27%	28–30%	overall
train	total	(1)	(4)	(26)	(28)	(44)	(28)	(19)	(15)	(10)	(175)
	top-1	0	0	2 8%	9 32%	25 57%	25 89%	13 68%	15 100%	9 90%	98 56%
	top-5	0 0%	0	4 15%	15 54%	32 73%	28 100%	17 89%	15 100%	10 100%	121 69%
	align	25/88 28%	143/421 34%	797/2153 37%	1564/2786 56%	3127/4423 71%	2704/3122 87%	1589/1743 91%	1022/1221 84%	886/979 91%	11857/16936 70%
test	total	(1)	(3)	(24)	(36)	(19)	(15)	(18)	(16)	(5)	(137)
	top-1	0	0	6 25%	18 50%	7 37%	11 73%	17 94%	16 100%	4 80%	79 58%
	top-5	0	0	9 37%	24 67%	10 53%	12 80%	18 100%	16 100%	5 100%	94 69%
	align	0/88 0%	116/312 37%	866/3023 29%	3384/5650 60%	1426/2392 60%	1673/2106 79%	3022/3412 89%	2091/2253 93%	489/505 97%	13067/19723 66%

Each column represents a different range of sequence identity level: *total* is the total number of pairs in a particular sequence-identity range; *top-1* and *top-5* represent the numbers of pairs whose templates are ranked in top one and top five, respectively; and *align* denotes the averaged alignment accuracy over all pair alignments in a particular sequence-identity range.

score. We have trained the neural network (Hertz et al. 1991) to accomplish such a mapping. To train a neural network, we need to provide a list of training vectors, each of which consists of an array of input data and a desired output value. In our case, the input data for each training vector consists of the following values: the total threading score, the pair contact potential, the singleton potential, the mutation energy, the gap penalty, and a series of "normalization" factors, including the lengths of the query and the template, the sequence identity between the query and the template in threading, and the (threading) scoring distribution between the template and the protein sequences of all the representative entries in FSSP (Holm and Sander 1996) (the actual input to the neural net includes the two extreme values and the average value of the distribution). One way to define the desired output value is simply to use 1 or 0, representing whether the template is the correct fold of the query or not, respectively, as Jones did in his recent work (1999).

We have used a different function to define a desired output value. For each query-template pair, we calculate the number of structurally alignable residues between them (note that all query proteins in our training set have 3D experimental structures in PDB) defined by the SARF program (Alexandrov 1996). Then for each query-template pair, the two proteins are considered to be the "true" pair if their FSSP indices share the same first digit (Holm and Sander 1996), that is, they belong to the same fold family. The frequency of the true pairs within a range of alignable residues provides a continuous function of how probable it may correspond to a true query-template pair if measured by structure-structure alignments alone. For example, on our dataset of 17,000+ false pairs and 708 true pairs, this number has the distribution shown in table 18.3. We found that using the relative frequencies listed as the desired outputs for neural net training has generated better normalization results (than the 0/1 output value) on our training set. This is probably because they provide more information than simply 0 or 1 to the neural net. Detailed discussion about the strength and properties of this training function is described by Xu et al. (2001).

Our neural net consists of 14 input nodes, one output node, and one hidden layer with 20 nodes. Among the entire data set (more than 18,000 pairs in total), one-half is used as the training set and the remaining half as the (independent) testing set. The goal of the training is to find a mathematical function that maps the input data to

Table 18.3
Relative frequency of true query-template pairs

Number of alignables	0–9	10–19	20–29	30–39	40–49	50–59	60–69	70–79	80–89	90–
Frequency being a true pair	0.000	0.001	0.003	0.020	0.150	0.500	0.750	0.900	0.990	1.000

values in the range of [0,1], which are as close to the desired output values as possible. Similar prediction results are achieved on the training and the testing sets, both having an error rate of about 11 percent. Table 18.4 summarizes the prediction performance on the combined set of training and testing.

This normalization by neural net has clearly given a meaning to each (normalized) score. For example, if a (normalized) score is between 0.6 and 0.7, we know that the probability of having a true fold recognition is about 0.83. Our preliminary test shows that ranking templates using normalized scores improves the accuracy of fold recognition, compared with using raw scores.

18.3.4 Using PROSPECT for Structure Prediction

PROSPECT provides a suite of options for various applications. It also has an interface to MODELLER (Sali et al. 1994) for generating atomic structures after a sequence-structure alignment has been constructed. It currently uses two template libraries that a user can choose: protein chains defined by the FSSP non-redundant set (Holm and Sander 1996), and compact domains defined by the DALI non-redundant domain library (Holm and Sander 1998). A detailed manual of PROSPECT is at http://compbio.ornl.gov/structure/prospect/. A Web server using supercomputers is also available at http://compbio.ornl.gov/structure/prospect_server/.

We now use the CASP-3 target t0057 (the CbiK protein) as a prediction example of applying the PROSPECT program. To obtain a template for t0057, we ran PROSPECT against the database of more than two thousand protein templates. The template 1ak1 was identified. Then we refined the alignment between t0053 and the template 1ak1 using the information of an active site. Through a detailed analysis, we found that 1ak1 has an active site at His-183 (Al-Karadaghi et al. 1997). Searching the BLOCK database (Henikoff and Henikoff 1994), we identified His-145 of t0053 as the corresponding active site. We then re-ran PROSPECT using the constraint that His-145 of t0053 should be aligned with His-183 of 1ak1. The final alignment between t0053 and 1ak1 (with a sequence identities of 12.2 percent) is given in figure 18.5. The refined alignment agrees quite well with the structure-structure alignment,

Table 18.4
Prediction performance on combined set of training and testing

Neural net output	0–0.1	0.1–0.2	0.2–0.3	0.3–0.4	0.4–0.5	0.5–0.6	0.6–0.7	0.7–0.8	0.8–0.9	0.9–1.0
No. of false pairs	17279	474	143	56	25	7	7	2	0	0
No. of true pairs	314	135	60	51	44	38	34	14	12	8
Percent of true pairs	1.7%	22%	30%	48%	64%	84%	83%	87%	100%	100%

```
The full alignment with 264 residues in the query sequence

MKKALLVVSFGTSYHDTCEKNIVACERDLAASCPDRD-L--FRA-F--TSGMI-IRKLRQ
 |  |||:..||.|.:       .  |      |: : |   :  .    |.  :  : .
-KMGLLVMAYGTPYKEEDIERYYTHIRR--GRKPEPEMLQDLKDRYEAIGGISPLAQITE

RDGIDIDTPLQAL-QKLAAQGYQDV------AIQSLHII--NG-DEYEKIV---------
    .::  |  :   :.   |  .         ..  .  .|  |   ||
QQAHNLEQHLNEIQDEITFKAYIGLKHIEPFIEDAVAEMHKDGITEAVSIVLAPHFSTFS

--REVQLLRPLFTRL-TLGVPLLSSHNDYVQLMQALRQQMPSLR--------QTEKVVFM
        :    :|  | .. .|  |    :      :            .   ..
VQSYNKRAKEEAEKLGGLTITSVESWYDEPKFVTYWVDRVKETYASMPEDERENAMLIVS

GH-------GASHHAFA-AYACLDHMMTAQRFP-ARVGAV------ES-Y-PEVDILIDS
 |                       .      .   ||        :.   |:|  |
AHSLPEKIKEFGDPYPDQLHESAKLIAEGAGVSEYAVGWQSEGNTPDPWLGPDVQDLTRD

L-RDEGVTGVHLMP-------L-MLVAGDHAINDMASDDGDSWKMRFNAAGIPATPWLSG
|    |      :|       | :|   |     :. | | |:            | ..
LFEQKGYQAFVYYVPVGFVADHLEVLYDNDYECKVVTDDIGASYY----------RPEMP-

LGENPAIRAMFVAHLHQALNMAVEEAA
    |       . .   |
-NAKPEFIDALATVVLKKLG-------
```

Figure 18.5
Sequence alignment between t0053 and 1ak1, where |, :, and . *indicate the two aligned residues are identical, similar, as related, respectively.*

and it is significantly better than the original one without the constraint. The structure using MODELLER based on the refined alignment is shown in figure 18.6. A detailed description of our prediction experience using PROSPECT in CASP-3 appears in Xu et al. 1999.

18.4 Application of Experimental Data as Threading Constraints

Though threading often provides valuable information, the computational structure models are still predictions. One way to significantly improve the reliability of predictions is through the incorporation of partial experimental data related to protein structures. Two classes of data have proven to be particularly useful in the pursuit of combined methods of experiments and computation for protein structure determination: (1) intramolecular cross-links from mass spectrometry, and (2) chemical shifts and long-range NOEs from NMR experiments. Potentially, these experimental pro-

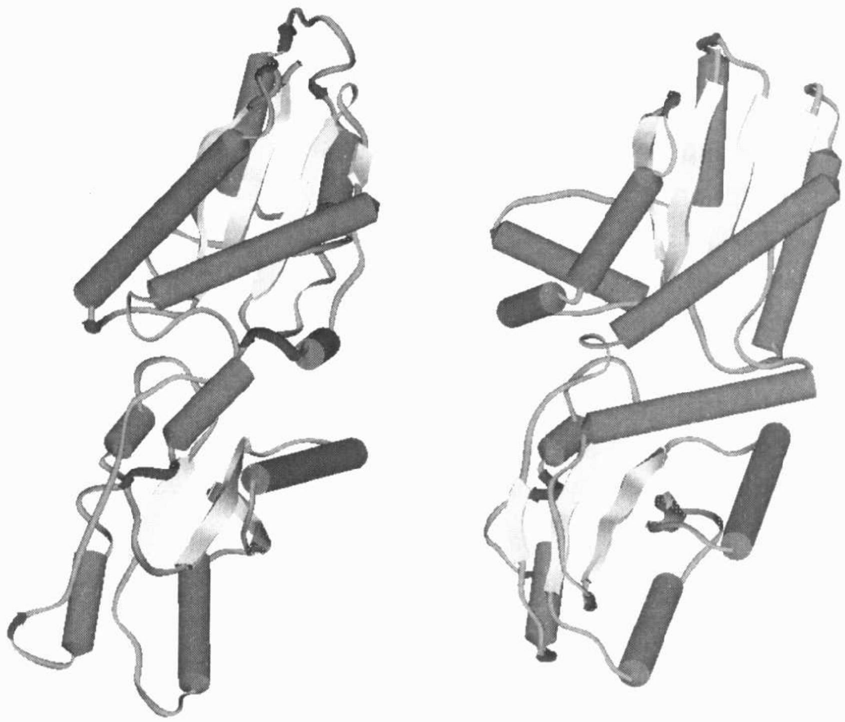


Figure 18.6
A comparison between the predicted structure (left) based on the template 1ak1 and the experimental one (right) for the CASP-3 target t0053. The target t0053 and the template 1ak1 belong to the same superfamily. The cylinders represent α-helices, the strands denote β-strands, the dark lines are for turns, and the thin lines represent loops.

cedures could produce structural data, not necessarily sufficient for an accurate structure determination, in a high-throughput mode. Our experience has shown that even a small number of distance restraints from NMR or cross-links could significantly reduce the search space size of *feasible* threading alignments, quickly ruling out the vast majority of the wrong folds and incorrect sequence-structure alignments. As the number of such distance restraints increases, we have observed that the quality of the predicted structures rapidly approaches the ones of the experimental structures. We could foresee the emergence of a new paradigm for protein structure determination in the near future, which will fully utilize the structural information derived from both the experiments and sophisticated computational procedures, making the structure determination process much more efficient and cost effective.

18.4.1 Intramolecular Cross-Links and Threading

A recent study by Young et al. (2000) demonstrates the effectiveness of using lysine (Lys-Lys) cross-links as constraints on threading for high-throughput protein fold

recognition. The basic idea of this approach is to probe a protein structure with a lysine-specific cross-linking agent, bis(sulfosuccinimidyl) suberate. Then a few cross-linked sites can be determined by tryptic peptide mapping using time-of-flight mass spectrometry. The distance between a pair of cross-linked lysines is bounded by the length of the cross-linking agent, that is, 24 Å between C_α-C_α atoms. The distance restraints are then used to assess the quality of threading models—how consistent the predicted structure models are with the obtained distance measures. The model that is most consistent with the cross-links is then chosen. Young et al. (2000) used 18 unique lysine cross-links, in conjunction with threading, and correctly identified FGF-2 as a member of the β-trefoil fold family. The structure model has a backbone error of 4.8 Å rmsd. This method is fast and uses a small amount of material. It is easy for automation and should be broadly applicable, so it has a potential for the high-throughput fold recognition.

18.4.2 Use of NMR Partial Data as Threading Constraints

The NMR method for protein structure determination is based on (1) proton and heteronuclear chemical shifts, (2) a network of distance restraints between spatially close ($\leq$ 5 Å) hydrogen atoms derived from nuclear Overhauser effects (NOEs), (3) dihedral angle restraints calculated from scalar coupling constants, and so on. An NMR structure is generally determined through molecular dynamics simulation and energy minimization under the constraints specified by NMR restraints (Braun and Gō 1985; Levy et al. 1989; Brünger 1992; Karimi-Nejad et al. 1998). It typically requires 15–20 NOE restraints per residue to obtain an accurate (mean) structure (equivalent to a ~2 Å X-ray structure).

It is well known that NMR methods work only for small proteins and that the effectiveness goes down rapidly as the weight of a protein goes beyond 30 kD. Of the 1,901 NMR structures in PDB (March 2000 release) (Bernstein et al. 1977), only 27 are larger than two hundred amino acids. The largest NMR structure in PDB is intimin (PDB code 1inm), with 282 residues (30.1 kD). The main problems with larger proteins are the increased number of resonances and line broadening, which result in spectral crowding and reduction in the fraction of spectral peaks that can be identified and assigned. Although NMR experiments generally cannot produce sufficient data for 3D structure determination of large proteins, it is often possible to get some NMR data (Lin and Wagner 1999). These partial NMR data have turned out to be highly valuable as constraints on threading-based fold recognition and backbone structure prediction, as we have recently demonstrated (Xu et al. 2000b, c). Particularly, we have found that (1) the topological information provided by just a few long-range NOEs (NOEs between residues that are not adjacent in the protein

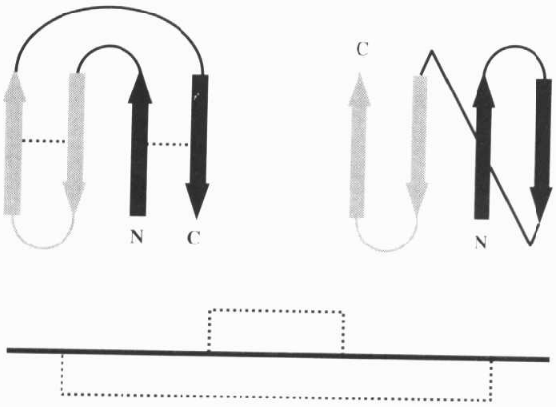


Figure 18.7
Ruling out false topological classes of proteins by using a few NOEs. On the left is a Greek key substructure, and on the right is a 4-antiparallel (flat) β-strand motif. A partial sequence with two NOEs is given in the bottom. Although an alignment between the Greek key structure and the partial sequence may exist that does not violate the NOE restraints, there is no such alignment between the 4-antiparallel β-strand motif and the sequence because it has the "wrong" topology.

sequence) can quickly rule out certain false fold classes and alignments during threading, as shown in figure 18.7; and (2) the geometric patterns of a group of NOEs, as shown in figure 18.8, can rapidly narrow down the list of potential alignments between a query protein and a template structure in threading.

We can define an *NOE-constrained threading problem* as to find a globally optimal threading (just as defined in section 18.2) between a query sequence and a template structure, under the constraint that no NOEs are "significantly" violated; that is, two residues with an NOE between them should not be aligned to template positions that are "too far" from each other. This can be implemented using a penalty function that penalizes threading alignments with "significant" deviations from the NOE-specified distances. We can treat structural information from other types of NMR data in a similar fashion.

The following gives a case study on how long-range NOEs affect the threading accuracy in both fold recognition and threading alignments. For this study, we have selected from the FSSP database (Holm and Sander 1996) 17 proteins as the queries. These proteins were selected randomly under the following conditions: (1) the query proteins have experimental NMR data in PDB; (2) they should evenly represent three classes of proteins: all-α, all-β, and α and β mixed; and (3) each has a native-like structure in the template set, with sequence identity less than 35 percent. In this test, we first ran the NMR-constrained threading program, and then ran MODELLER (Sali et al. 1994) to generate detailed 3D structures based on threading alignments without using additional data. The fold recognition study is done against 667 unique folds.

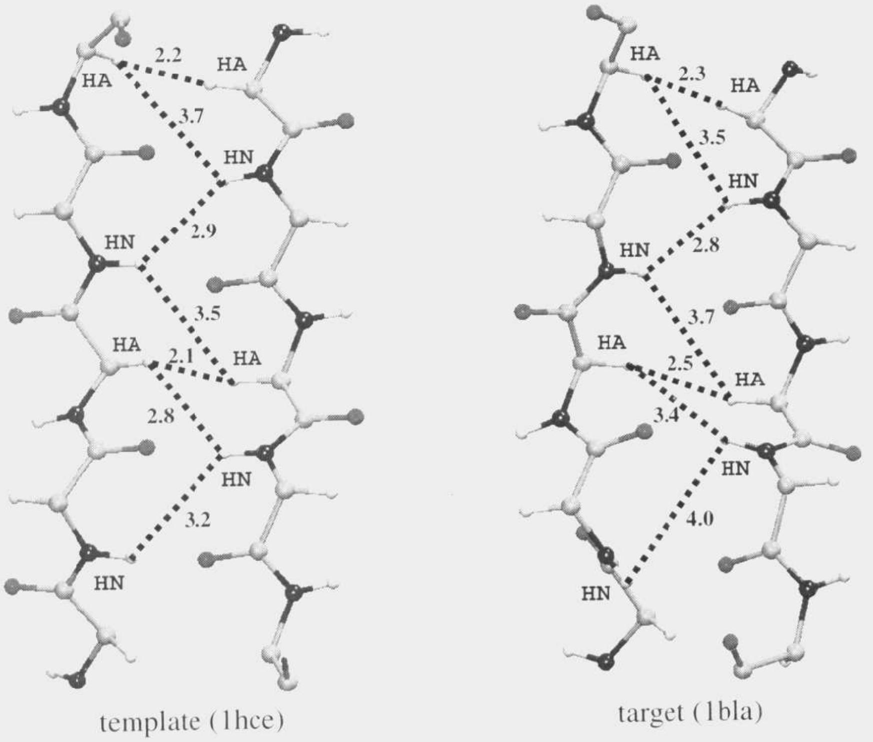


Figure 18.8
Comparison between the structurally equivalent β-sheets in the two proteins of similar folds. Residues 74–78 and 103–107 for template 1hce, and residues 83–87 and 113–117 for the query protein 1bla, are shown. The dashed lines with the numbers give the distance in Å.

We tested how the amount of NOE information affects the threading performance, using 0, 0.5, 1.0, 1.5, 2.0, 2.5, and 3.0 NOEs (including both long- and short- NOEs) per residue, respectively. The NOEs are selected randomly and uniformly from the NMR data file of the corresponding protein. Table 18.5 summarizes the test results. The tabulated alignment accuracy between structurally equivalent residues in the query and template (C_α-RMSD) is the highest accuracy in ten runs.[4] The fold recognition accuracy is based on a single run.

In this test, NOEs improve the threading performance in 13 out of the 17 cases, with seven cases showing improvements in fold recognition and 11 cases showing improvements in alignment accuracy. One observation we have is that although the pure threading results on all three analogous pairs (1afi-2acy, 3phy-1bv6, and 1bla-1hce, as classified by SCOP [Murzin et al. 1995]) are generally poor and worse than those of homologous pairs, the prediction results with at least one NOE per residue

4. Each run represents a different sampling of NOEs given a fixed number of NOEs per residue.

Table 18.5
Threading accuracy vs. the number of NOE restraints

Class	Query	nres (a.a.)	temp	iden (%)	rmsd (Å)	RMSD (Å)/rank vs. NOE/a.a.						
						0	0.5	1.0	1.5	2.0	2.5	3.0
α	1bbn	133	1cnt1	8	2.6	**16.2/29**	6.6/15	6.6/1	6.6/2	5.3/1	5.3/1	**5.3/1**
	1itl	130	3inkC	12	2.2	**13.1/7**	4.3/26	4.2/1	4.2/4	3.6/3	3.6/3	**3.6/3.**
	1ner	74	1lmb3	14	2.4	3.8/**57**	3.8/57	3.8/57	3.8/79	3.8/49	3.8/39	3.8/**39**
	1il6	166	1bgc	15	2.1	3.0/1	3.0/1	3.0/1	3.0/1	3.0/1	3.0/1	3.0/1
	1ocd	104	1cyj	28	2.0	3.4/1	3.4/1	3.4/1	3.4/1	3.4/1	3.4/1	3.4/1
β	1maj	113	1agdB	7	2.5	**8.8/37**	8.0/10	6.9/1	6.9/1	6.6/1	6.6/1	**6.6/1**
	1nct	98	1bec	12	2.5	**14.7**/1	13.8/1	5.4/1	5.4/1	5.4/1	5.4/1	**5.4**/1
	1bla	155	1hce	14	2.1	**7.7**/1	4.0/1	4.0/1	4.0/1	4.0/1	4.0/1	**4.0**/1
	1vhp	117	1cd8	15	2.9	**4.5**/1	4.4/1	4.4/1	4.4/1	4.4/1	4.4/1	**4.4**/1
	1ghj	79	1iyu	24	1.8	2.0/1	2.0/1	2.0/1	2.0/1	2.0/1	2.0/1	2.0/1
	1a7i	60	1qli	33	2.5	**2.8**/1	2.8/1	2.8/1	2.6/1	2.6/1	2.6/1	**2.6**/1
α/β	1afi	72	2acy	7	2.8	**6.8/376**	5.5/193	5.6/26	5.5/20	3.8/3	3.8/2	**3.8/2**
	3trx	105	1a8y	12	2.7	**4.1**/1	3.1/1	3.1/1	3.1/1	3.2/1	3.0/1	**3.0/1**
	1ikm	69	1dokB	13	2.1	2.1/**2**	2.1/2	2.1/2	2.1/1	2.1/1	2.1/1	2.1/**1**
	3phy	125	1bv6	15	2.3	**12.2/41**	8.8/25	3.6/31	3.6/5	3.6/6	3.6/6	**3.6/4**
	1erp	166	1byuB	24	1.7	2.6/1	2.6/1	2.6/1	2.6/1	2.6/1	2.6/1	2.6/1
	1fht	116	2u1a	34	2.1	**4.5**/1	4.5/1	4.5/1	4.5/1	4.5/1	4.5/1	**2.1**/1

Query and *temp* represent the query and template proteins, respectively. *Nres* is the number of alignable residues between the query and template. *Iden* denotes the sequence identity between the query and template sequences. The entry *rmsd* is the C_γ-RMSD between the structurally equivalent residues of the query and template structures. *RMSD rank vs. NOE a.a.* represents (a) the C_γ-RMSD between the experimental structure and the predicted structure and (b) the rank of the correct template structure among all templates, respectively, when using 0, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0 (long- and short-range) NOEs per residue as threading constraints.

for the analogous pairs and the remaining 14 homologous pairs are quite comparable. This seems to suggest the possibility of reliably extending the scope of threading from homologs to analogs, with the help of a few NOEs.

18.5 Discussion

18.5.1 Implications of Threading for the Structural Genomics Initiative

Traditionally, protein structures were solved at a low-throughput mode, one protein at a time. Recent advances in synchrotron and high-resolution NMR have substantially accelerated the rate of protein structure determination. There is an overwhelming consensus in the structural biology community that protein structures can be solved massively (an effort called "structural genomics"), in a similar fashion to how DNA sequences are being sequenced. The goal of the recent NIH Structural Genomics Initiative (National Institute of General Medical Sciences 1999) is to determine the structures of a hundred thousand human proteins. The potential impact of such a project could be as significant as the Human Genome Project. The basic strategy of the project is to first select a representative from each fold family and solve them using experimental means including X-ray crystallography and NMR, and then to computationally model the rest of the structures using the representative structures as templates. This would save a tremendous amount of time and resources if successful. Two key computational challenges in this ambitious initiative are (1) to identify which protein sequences represent novel folds, and (2) to computationally model the rest of the structures based on the representative structures. Clearly threading could play a key role in addressing these challenges.

Several groups have employed threading tools to structurally model all coding sequences from a whole genome (Fischer and Eisenberg 1997; Jones 1999). These studies have provided timely analysis for the current genome sequencing efforts, and allow gene-hunting researchers to find valuable structure information quickly. It has been found that about 17 percent of the identified genes fall into the category of homology modeling (Sanchez and Sali 1998). The sequence profile approach extends the genome-wide coverage from 17 percent to about 30–40 percent (Gerstein 1998). The threading approach can predict about an additional 10–20 percent of identified genes with good confidence (Jones 1999). For the rest of the proteins, either existing threading methods cannot recognize the templates with good confidence, or templates do not exist in the structure database (i.e., novel folds). This raises two challenging issues—how to further improve the efficacy of threading methods, and how to reliably determine if a protein has a new fold? Among some of the proteins with new folds is the class of membrane proteins, which have not been touched by threading

methods. Due to the technical difficulty of experimentally solving these structures and the lack of available templates for computational methods like threading, it may take a long time to solve all these proteins. However, with the advancement of new experimental techniques and computational methods, it is foreseeable to have the structures of all globular proteins solved within the next ten years, and hence put an end to the folding problem of globular proteins.

18.5.2 Challenging Issues and New Developments

Although about 90 percent of new globular proteins may have native-like folds in the structure database, as we discussed in section 18.2, current threading results clearly fall short of these theoretical limits (CASP 1995, 1997, 1999). Among all the prediction targets in CASP-3, 25 proteins were considered as "solvable" by protein threading and were put in the category of fold recognition. Only 13 of the 25 proteins had their correct folds recognized and had at least 50 percent of the alignable residues aligned correctly by any participating team (Marchler-Bauer et al. 1999). Further improving the accuracy and enlarging the scope of protein threading are among the key challenging issues in computational structural biology.

Local Threading Virtually all existing threading programs attempt to solve the "global" threading problem, finding an optimal alignment between the whole sequence of a query protein and the entire structure of a template protein. This type of threading method is not very sensitive in detecting templates when the percentage of structurally alignable residues is low (say, <50%)—existing threading methods work well only when the query protein has a template in PDB with at least 60–70 percent of their residues structurally alignable (Marchler-Bauer et al. 1999). This has clearly limited the applicability of protein threading.

One way to solve this problem is through "local threading", locating good partial sequence-structure alignments (e.g., a structure domain or a super-secondary structure). This will extend the scope of current threading methods to deal with a much larger class of structure prediction problems. Local threading may provide a better way to recognize function-related structural motifs (for example, the EF-hand motif frequently found in calcium binding proteins with various folds) than sequence-based motif search methods. However, as we know, many of the motifs do not have strong sequence characteristics, so sequence-based motif search methods may not work well.

Mini-Threading Threading takes advantage of enormous structure information in PDB and provides more reliable structure predictions than ab initio methods. However, threading does not apply to the proteins that do not have templates in PDB or have templates that threading cannot recognize. On the other hand, ab initio pre-

diction applies to all proteins. However, it generally requires a significant amount of computing resource, and its prediction results might be unreliable. Mini-threading (Simons et al. 1997) attempts to combine the strengths of these two approaches. It first builds local structures through sequence comparison or local threading, and then assembles them into a global structure through ab initio methods. Current mini-threading methods typically recognize structure segments in PDB that are most similar to a segment of the query sequence (with a typical size of nine amino acids) in terms of sequence similarity. Then a distribution profile is constructed for the ϕ and ψ angles of the query protein's backbones, based on the distributions in the structure segments. It is possible to use templates that are fine-tuned for small structure segments like *I-sites* (Bystroff and Baker 1998) to obtain more reliable local structure templates. Once local structures are more or less defined, assembling them based on their backbone distributions would require significantly fewer computational resources than pure ab initio methods. The optimization process is typically carried out using genetic algorithms (Unger and Moult 1992) or Monte Carlo simulations (Skolnick and Kolinski 1991). Some success of mini-threading has been demonstrated in CASP-3 (CASP 1999). Mini-threading provides a promising method to generate low resolution structures for the proteins to which threading might not apply.

Using Multiple-Sequence or Multiple-Structure Information One of the problems in threading is the structural variations among proteins of the same fold, which is not well captured in the existing threading scoring functions. When two proteins share the same fold, the length of the structurally alignable secondary structures and the packing between the secondary structures may differ, particularly when the two proteins are structural analogs (in the same fold but not the same superfamily), where the variations in the secondary structures of peripheral elements as well as in some core elements are much larger than structural homologs. The solvent accessibilities, the secondary structure types, and the C_β pairs of the template may not correspond exactly to those of the query protein in the aligned positions.

One way to overcome the problem is to derive the basic statistics (used to calculate the scoring functions) from the "consensus" of the query sequence or template fold. One can generate a sequence profile for the query's protein family based on multiple-sequence alignments. Such a profile reflects common characteristics of the protein family, and hence it can enhance threading sensitivity while reducing noises. Another way to use protein classification is to construct the sequence profile for the protein family containing the fold template or to build profile for the structures of the same fold through multiple structure alignments (Panchenko et al. 2000). By using the structure alignment, one can identify which positions (as well as solvent accessibilities,

the secondary structure types, or the C_β pairs) are more conserved across different templates of the same fold. The advantage of using multiple templates instead of multiple sequences in the query's family is that the former covers the profile of proteins of the same fold (not necessarily of the same family or superfamily).

18.5.3 Future Outlook

Protein threading is becoming a major tool in computational biology. Its applicability will probably cover all globular proteins within the next decade. We will continue to see improvement in the prediction accuracy and computational efficiency of threading-based structure prediction methods. Genome-scale application of threading will become routine in the next few years. As technologies mature in high-throughput experimental and computational structural biology, we will see a merger of the two approaches. Information from structural experiments and template-based computation will be tightly combined for significantly more efficient and cost-effective ways of determining protein structures.

Acknowledgments

This work was supported by the Office of Biological and Environmental Research, U.S. Department of Energy, under Contract DE-AC05-00OR22725, managed by UT-Battelle, LLC.

References

Abagyan, R. A., and Batalov, S. (1997). Do aligned sequences share the same fold? *J. Mol. Biol.* 273: 355–368.

Al-Karadaghi, S., Hansson, M., Nikonov, S., Jonsson, B., and Hederstedt, L. (1997). Crystal structure of ferrochelatase: The terminal enzyme in heme biosynthesis. *Structure* 5: 1501–1510.

Alexandrov, N. N. (1996). SARFing the PDB. *Protein Eng.* 9: 727–732.

Alexandrov, N. N., Nussinov, R., and Zimmer, R. M. (1996). Fast protein fold recognition via sequence to structure alignment and contact capacity potentials. In *Biocomputing: Proceedings of the 1996 Pacific Symposium*, Hunter, L. and Klein, T., ed., 53–72. Singapore: World Scientific Publishing Co.

Altschul, S. F., Madden, T. L., Schäffer, A. A., Zhang, J., Zhang, Z., Miller, W., and Lipman, D. J. (1997). Gapped BLAST and PSI-BLAST: A new generation of protein database search programs. *Nucl. Acids Res.* 25: 3389–3402.

Bernstein, F. C., Koetzle, T. F., Williams, G. J. B., Meyer, E. F., Brice, M. D., Rodgers, J. R., Kennard, O., Shimanouchi, T., and Tasumi, M. (1977). The protein data bank: A computer based archival file for macromolecular structures. *J. Mol. Biol.* 112: 535–542.

Bowie, J. U., Luthy, R., and Eisenberg, D. (1991). A method to identify protein sequences that fold into a known three-dimensional structure. *Science* 253: 164–170.

Braun, W., and Gō, N. (1985). Calculation of protein conformations by proton-proton distance constraints: A new efficient algorithm. *J. Mol. Biol.* 186: 611–626.

Brooks, B. R., Bruccoleri, R. E., Olafson, B. D., States, D. J., Swaminathan, S., and Karplus, M. (1983). CHARMM: A program for macromolecular energy, minimization, and dynamics calculations. *J. Comp. Chem.* 4: 187–217.

Brünger, A. T. (1992). *X-PLOR, Version 3.1, A System for X-ray Crystallography and NMR*. New Haven: The Howard Hughes Medical Institute and Department of Molecular Biophysics and Biochemistry, Yale University.

Bryant, S. H., and Altschul, S. F. (1995). Statistics of sequence-structure threading. *Curr. Opinion Struct. Biol.* 5: 236–244.

Bryant, S. H., and Lawrence, C. E. (1993). An empirical energy function for threading protein sequence through the folding motif. *Proteins: Struct. Funct. Genet.* 16: 92–112.

Bystroff, C., and Baker, D. (1998). Prediction of local structure in proteins using a library of sequence-structure motifs. *J. Mol. Biol.* 281: 565–577.

CASP (1995). Protein structure prediction issue. *Proteins: Struct. Funct. Genet.* 23: 295–462.

CASP (1997). Protein structure prediction issue. *Proteins: Struct. Funct. Genet.* suppl. 1, 29: 1–230.

CASP (1999). Protein structure prediction issue. *Proteins: Struct. Funct. Genet.* suppl. 3, 37: 1–237.

Chothia, C. (1992). One thousand families for the molecular biologist. *Nature* 357: 543–544.

Crawford, O. H. (1999). A fast, stochastic algorithm for protein threading. *Bioinformatics* 15: 66–71.

Dayhoff, M. O. (1978). A model of evolutionary change in proteins. *Atlas of Protein Sequences and Structure* 5 (supplement 3): 345–352.

de las Alas, M. M., de Bruin, R. A., Ten Eyck, L., Los, G., and Howell, S. B. (1998). Prediction-based threading of the hMSH2 DNA mismatch repair protein. *FASEB J.* 12: 653–663.

Dorit, R. L., Schoenbach, L., and Gilbert, W. (1990). How big is the universe of exons? *Science* 250: 1377–1382.

Drexler, K. E. (1981). Molecular engineering: An approach to the development of general capabilities for molecular manipulation. *Proc. Natl. Acad. Sci. USA* 78: 5275–5258.

Finkelstein, A. V., and Ptitsyn, O. B. (1987). Why do globular proteins fit the limited set of folding patterns? *Prog. Biophys. Mol. Biol.* 50: 171–190.

Fischer, D., and Eisenberg, D. (1996). Fold recognition using sequence-derived predictions. *Protein Science* 5: 947–955.

Fischer, D., and Eisenberg, D. (1997). Assigning folds to the proteins encoded by the genome of *mycoplasma genitalium*. *Proc. Natl. Acad. Sci. USA* 94: 11929–11934.

Fischer, D., Elofsson, A., Bowie, J. U., and Eisenberg, D. (1996a). Assessing the performance of fold recognition methods by means of a comprehensive benchmark. In *Biocomputing: Proceedings of the 1996 Pacific Symposium*, Hunter, L. and Klein, T., eds., 300–318. Singapore: World Scientific Publishing Co.

Fischer, D., Rice, D., Bowie, J. U., and Eisenberg, D. (1996b). Assigning amino acid sequences to 3-dimensional protein folds. *FASEB J.* 10: 126–136.

Flockner, H., Braxenthaler, M., Lackner, P., Jaritz, M., Ortner, M., and Sippl, M. J. (1995). Progress in fold recognition. *Proteins: Struct. Funct. Genet.* 23: 376–386.

Friedrichs, M. S., and Wolynes, P. G. (1989). Towards protein tertiary structure recognition by means of associative memory Hamiltonians. *Science* 246: 371.

Gerstein, M. (1998). Patterns of protein-fold usage in eight microbial genomes: A comprehensive structural census. *Proteins: Struct. Funct. Genet.* 33: 518–534.

Godzik, A., Skolnick, J., and Kolinski, A. (1992). A topology fingerprint approach to the inverse folding problem. *J. Mol. Biol.* 227: 227–238.

Gonnet, G. H., Cohen, M. A., and Benner, S. A. (1992). Exhaustive matching of the entire protein sequence database. *Science* 256: 1443–1445.

Henikoff, S., and Henikoff, J. G. (1992). Amino acid substitution matrices from protein blocks. *Proc. Natl. Acad. Sci. USA* 89: 10915–10919.

Henikoff, S., and Henikoff, J. G. (1994). Protein family classification based on searching a database of blocks. *Genomics* 19: 97–107.

Hertz, J., Krogh, A., and Palmer, R. G. (1991). *Introduction to the Theory of Neural Computation*. New York: Addison-Wesley.

Holm, L., and Sander, C. (1996). Mapping the protein universe. *Science* 273: 595–602.

Holm, L., and Sander, C. (1998). Dictionary of recurrent domains in protein structures. *Proteins: Struct. Funct. Genet.* 33: 88–96.

Hu, X., Xu, D., Hamer, K., Schulten, K., Koepke, J., and Michel, H. (1995). Predicting the structure of the light-harvesting complex II of *Rhodospirillum molischianum*. *Protein Science* 4: 1670–1682.

Hughey, R., and Krogh, A. (1996). Hidden Markov models for sequence analysis: Extension and analysis of the basic method. *CABIOS* 12: 950–107.

Jones, D. T. (1999). GenTHREADER: An efficient and reliable protein fold recognition method for genomic sequences. *J. Mol. Biol.* 287: 797–815.

Jones, D. T., Taylor, W. R., and Thornton, J. M. (1992). A new approach to protein fold recognition. *Nature* 358: 86–89.

Karimi-Nejad, Y., Warren, G. L., Schipper, D., Brünger, A. T., and Boelens, R. (1998). NMR structure calculation methods for large proteins—application of torson angle dynamics and distance geometry/simulated annealing to the 269-residue protein serine protease PB92. *Mol. Phys.* 95: 1099–1112.

Karlin, S., and Altschul, S. F. (1990). Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes. *Proc. Natl. Acad. Sci. USA* 87: 2264–2268.

Karlin, S., Dembo, A., and Kawabata, T. (1990). Statistical composition of high-scoring segments from molecular sequences. *Ann. Statistics* 18: 571–581.

Krogh, A., Brown, M., Mian, I. S., Sjolander, K., and Haussler, D. (1994). Hidden Markov models in computational biology: Applications to protein modeling. *J. Mol. Biol.* 235: 1501–1531.

Lathrop, R. H. (1994). The protein threading problem with sequence amino acid interaction preferences in np-complete. *Protein Eng.* 7: 1059–1068.

Lathrop, R. H., and Smith, T. F. (1996). Global optimum protein threading with gapped alignment and empirical pair score functions. *J. Mol. Biol.* 255: 641–665.

Levitt, M., and Chothia, C. (1981). Structural patterns in globular proteins. *Nature* 261: 552–558.

Levitt, M., and Warshel, A. (1975). Computer simulation of protein folding. *Nature* 253: 694–698.

Levy, R. M., Bassolino, D. A., Kitechen, D. B., and Pardi, A. (1989). Solution structures of proteins from NMR data and modeling: Alternative folds for neutrophil peptide 5. *Biochemistry* 28: 9361–9372.

Li, H., Helling, R., Tang, C., and Wingreen, N. (1996). Emergence of preferred structures in a simple model of protein folding. *Science* 273: 666–669.

Li, Z., and Scheraga, H. A. (1987). Monte Carlo-minimization approach to the multiple-minima problem in protein folding. *Proc. Natl. Acad. Sci. USA* 84: 6611–6615.

Lin, Y., and Wagner, G. (1999). Efficient side-chain and backbone assignment in large proteins: Application to tGCN5. *J. Biomol. NMR* 15: 227–239.

Lo Conte, L., and Smith, T. F. (1997). Visible volume: A robust measure for protein structure characterization. *J. Mol. Biol.* 273: 338–348.

Luthy, R., Bowie, J. U., and Eisenberg, D. (1992). Assessment of protein models with three-dimensional profiles. *Nature* 356: 83–85.

Madej, T., Gibrat, J. F., and Bryant, S. H. (1995). Threading a database of protein cores. *Proteins: Struct. Funct. Genet.* 23: 356–369.

Marchler-Bauer, A., Addess, K. J., Chappey, C., Geer, L., Madej, T., Matsuo, Y., Wang, Y., and Bryant, S. H. (1999). MMDB: Entrez's 3D structure database. *Nucl. Acids Res.* 27: 240–243.

Marchler-Bauer, A., and Bryant, S. H. (1997). A measure of success in fold recognition. *Trends in Biochemical Sciences* 22: 236–240.

Milburn, D., Laskowski, R. A., and Thornton, J. M. (1998). Sequences annotated by structure: A tool to facilitate the use of structural information in sequence analysis. *Protein Eng.* 11: 855–859.

Mohanty, D., Dominy, B. N., Kolinski, A., Brooks 3rd, C. L., and Skolnick, J. (1999). Correlation between knowledge-based and detailed atomic potentials: Application to the unfolding of the GCN4 leucine zipper. *Proteins: Struct. Funct. Genet.* 35: 447–452.

Murzin, A. G., Brenner, S. E., Hubbard, T., and Chothia, C. (1995). Scop: A structural classification of proteins database for the investigation of sequences and structures. *J. Mol. Biol.* 247: 536–540.

National Institute of General Medical Sciences (1999). Pilot projects for the protein structure initiative (structural genomics). http://www.nih.gov/grants/guide/arffiles/RFA-GM-99-009.html, June: RFA GM 99–009.

Needleman, S. B., and Wunsch, C. D. (1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J. Mol. Biol.* 48: 443–453.

Nilges, M., and Brünger, A. (1993). Successful prediction of the coiled coil geometry of the gcn4 leucine zipper domain by simulated annealing: Comparison to the x-ray structure. *Proteins: Struct. Funct. Genet.* 15: 133–146.

Orengo, C. A., Jones, D. T., and Thornton, J. M. (1994). Protein superfamilies and domain superfolds. *Nature* 372: 631–634.

Orengo, C. A., Michie, A. D., Jones, S., Jones, D. T., Swindells, M. B., and Thornton, J. M. (1997). CATH: A hierarchic classification of protein domain structures. *Structure* 5: 1093–1108.

Pabo, C. (1983). Molecular technology. Designing proteins and peptides. *Nature* 301: 200–200.

Panchenko, A. R., Marchler-Bauer, A., and Bryant, S. H. (2000). Profiles based on structure alignments increase the sensitivity of protein threading. In *Quantitative Challenges in the Post-Genome Sequence Era: a Workshop and Symposium*, The La Jolla Interfaces in Science, 2, La Jolla, CA.

Pedersen, J. T., and Moult, J. (1997). Protein folding simulations with genetic algorithms and a detailed molecular description. *J. Mol. Biol.* 269: 240–259.

Peitsch, M. C. (1996). ProMod and Swiss-Model: Internet-based tools for automated comparative protein modelling. *Biochem. Soc. Trans.* 24: 274–279.

Ponder, J. W., and Richards, F. M. (1987). Tertiary templates for proteins. Use of packing criteria in the enumeration of allowed sequences for different structural classes. *J. Mol. Biol.* 193: 775–791.

Rost, B. (1995). TOPITS: Threading one-dimensional predictions into three-dimensional structures. *ISMB* 3: 314–321.

Rost, B., and Sander, C. (1993). Prediction of protein secondary structure at better than 70% accuracy. *J. Mol. Biol.* 232: 584–599.

Sali, A., and Blundell, T. L. (1993). Comparative protein modelling by satisfaction of spatial restraints. *J. Mol. Biol.* 234: 779–815.

Sali, A., Shakhnovich, E., and Karplus, M. (1994). Kinetics of protein folding. A lattice model study of the requirements for folding to the native state. *J. Mol. Biol.* 235: 1614–1636.

Sanchez, R., and Sali, A. (1998). Large-scale protein structure modeling of the *saccharomyces cerevisiae* genome. *Proc. Natl. Acad. Sci. USA* 95: 13597–13602.

Simons, K. T., Kooperberg, C., Huang, E., and Baker, D. (1997). Assembly of protein tertiary structures from fragments with similar local sequences using simulated anealing and Bayesian scoring functions. *J. Mol. Biol.* 268: 209–225.

Sippl, M. J., and Weitckus, S. (1992). Detection of native-like models for amino acid sequences of unknown three-dimensional structure in a data base of known protein conformations. *Proteins: Struct. Funct. Genet.* 13: 258–271.

Skolnick, J., and Kolinski, A. (1991). Dynamic Monte Carlo simulations of a new lattice model of globular protein folding, structure and dynamics. *J. Mol. Biol.* 221: 499–531.

Smith, T. F., Conte, L. L., Bienkowska, J., Gaitatzes, C., Rogers, R., and Lathrop, R. (1997). Current limitations to protein threading approaches. *J. Comp. Biol.* 4(3): 217–225.

Smith, T. F., and Waterman, M. S. (1981). Comparison of biosequences. *Adv. Appl. Math.* 2: 482–489.

Srinivasan, B. N., and Blundell, T. L. (1993). An evaluation of the performance of an automated procedure for comparative modelling of protein tertiary structure. *Protein Eng.* 6: 501–512.

Unger, R., and Moult, J. (1992). Potential of genetic algorithms in protein folding and protein engineering simulations. *J. Mol. Biol.* 5: 637–645.

Villoutreix, B. O., Blom, A. M., and Dahlback, B. (1999). Structural prediction and analysis of endothelial cell protein C/activated protein C receptor. *Protein Eng.* 12: 833–840.

Wang, Z. X. (1996). How many fold types of protein are there in nature? *Proteins: Struct. Funct. Genet.* 26: 186–191.

Wang, Z. X. (1998). A re-estimation for the total numbers of protein folds and super-families. *Protein Eng.* 11: 621–626.

Xu, D., Unseren, M. A., Xu, Y., and Uberbacher, E. C. (2000a). Protein fold recognition at the secondary structure level. *Bioinformatics* 16: 257–268.

Xu, Y., and Xu, D. (2000). Protein threading using PROSPECT: Design and evaluation. *Proteins: Struct. Funct. Genet.* 40: 343–354.

Xu, Y., Xu, D., Crawford, O. H., and Einstein, J. R. (2000b). A computational method for NMR-constrained protein threading. *J. Comp. Biol.* 7: 449–467.

Xu, Y., Xu, D., Crawford, O. H., Einstein, J. R., Larimer, F., Uberbacher, E. C., Unseren, M. A., and Zhang, G. (1999). Protein threading by PROSPECT: A prediction experiment in CASP-3. *Protein Eng.* 12: 899–907.

Xu, Y., Xu, D., Crawford, O. H., Einstein, J. R., and Serpersu, E. (2000c). Protein structure determination using protein threading and sparse NMR data. In *The Fourth Annual International Conference on Computational Molecular Biology*, Shamir, R., Miyano, S., Istrail, S., Pevzner, P., and Waterman, M., eds., 299–307. Tokyo: ACM.

Xu, Y., Xu, D., and Olman, V. N. (2001). A practical method for interpretation of threading scores: An application of neural network. *Statistica Sinica* special issue on bioinformatics, in press.

Xu, Y., Xu, D., and Uberbacher, E. C. (1998a). An efficient computational method for globally optimal threading. *J. Comp. Biol.* 5(3): 597–614.

Xu, Y., Xu, D., and Uberbacher, E. C. (1998b). A new method for modeling and solving the protein fold recognition problem. In *The Second Annual International Conference on Computational Molecular Biology*, Istrail, S., Pevzner, P., and Waterman, M., eds., 285–292. New York: ACM.

Young, M. M., Tang, N., Hempel, J. C., Oshiro, C. M., Taylor, E. W., Kuntz, I. D., Gibson, B. W., and Dollinger, G. (2000). High throughput protein fold identification by using experimental constraints derived from intramolecular cross-links and mass spectrometry. *Proc. Natl. Acad. Sci. USA* 97: 5802–5806.

Zhang, B., Jaroszewski, L., Rychlewski, L., and Godzik, A. (1997). Similarities and differences between nonhomologous proteins with similar folds: Evaluation of threading strategies. *Folding and Design* 2: 307–317.

Zhang, C., and DeLisi, C. (1998). Estimating the number of protein folds. *J. Mol. Biol.* 284: 1301–1305.

Zimmer, R., Wohler, M., and Thiele, R. (1998). New scoring schemes for protein fold recognition based on voronoi contacts. *Bioinformatics* 14: 295–308.

Appendix. Web Addresses of Protein Threading Programs

123D	http://cartan.gmd.de/ToPLign.html	server
123D+	http://www-lmmb.ncifcrf.gov/~ nicka/123D+.html	server
3D-PSSM	http://www.bmm.icnet.uk/servers/3dpssm/	server
bioinbgu	http://www.cs.bgu.ac.il/~bioinbgu/	server
FFAS	http://bioinformatics.burnham-inst.org/FFAS/	server
FUGUE	http://www-cryst.bioc.cam.ac.uk/~ fugue/prfsearch.html	server
GenTHREADER	http://www.psipred.net	server
loopp	http://ser-loopp.tc.cornell.edu/loopp.html	server
NCBI Package	http://www.ncbi.nlm.nih.gov/Structure/	executables
PROFIT	http://lore.came.sbg.ac.at/	executables
PROSPECT	http://compbio.ornl.gov/structure/prospect_server/	server
PROSPECTOR	http://bioinformatics.danforthcenter.org/services/threading.html	server
SAS	http://www.biochem.ucl.ac.uk/bsm/sas/	server
Sausage	http://rsc.anu.edu.au/~arussell/TheSausageMachine.html	server
TOPITS	http://dodo.cpmc.columbia.edu/predictprotein/	server
ToPLign	http://cartan.gmd.de/ToPLign.html	server
UCLA-DOE	http://www.doe-mbi.ucla.edu/people/frsvr/frsvr.html	server

19 Computational Methods for Docking and Applications to Drug Design: Functional Epitopes and Combinatorial Libraries

Ruth Nussinov, Buyong Ma, and Haim J. Wolfson

19.1 Introduction

Several ingredients are needed in order to efficiently and successfully search a library of inhibitors, or drugs, with the goal of optimally docking them onto a specific target receptor (Nussinov and Wolfson 1999a, b): first, an adequate molecular surface representation; second, efficient docking techniques; third, a practical way of accounting for molecular surface variability; and fourth, providing for molecular flexibility. These four ingredients yield the candidate molecules. The fifth critical component is a fast, empirical way of scoring the large number of obtained solutions and ranking them. Currently, although there exist a variety of computational docking approaches, the scoring step has proven to be the most difficult hurdle.

Prediction of the docked conformation of a receptor-ligand molecule pair without any additional knowledge as to their binding sites is an extremely complex problem (e.g., Goodsell and Olson 1990; Bacon and Moult 1992; Kuntz et al. 1982; Connolly 1986; Cherfils et al. 1991; Katchalsky-Katzir et al. 1992; Jiang and Kim 1991; Shoichet and Kuntz 1991; Wang 1991; Cherfils and Janin 1993; Norel et al. 1994a, b; Helmer-Citterich and Tramontano 1994; Fischer et al. 1995; Norel et al. 1995; Lengauer and Rarey 1996; Wallqvist and Covell 1996; Rarey et al. 1996; Jones et al. 1997; Kasinos et al. 1992; Gabb et al. 1997). The problem can be defined as follows: Given the atomic coordinates of the two molecules, predict their native bound association. Clearly, in principle, every portion of surface of one molecule should be matched with every portion of the other, in all rotations and translations. The number of possible matched configurations is immense. Further, even if the binding site can be predicted (via, for example, its being the largest cavity, e.g., Peters et al. 1996; Laskowski et al. 1996), there is no guarantee that the corresponding candidate trial ligand will not bind at other, alternate sites. A particularly severe complication is the molecular surface variability. In solution, the surfaces of the molecules are in constant motion. Movements of side chains and surface atoms implicitly forces taking account of intermolecular penetrations of the docked molecule pair. In solution, such surface penetrations are alleviated by the movements of the groups of atoms on the molecular surface (Norel et al. 1998, 1999a). By contrast, obviously, allowing full-fledged molecular flexibility, that is, allowing every two atoms to move with respect to each other while we search through databases of molecules is entirely infeasible. Thus, we need to devise some practical approaches in which some degree of flexibility will still be permitted. Moreover, in addition to molecular surface variability, one needs to

consider domain motions. Docking rigid molecules may miss the correct solutions altogether.

Below we describe efficient computational approaches to the docking problem. We focus on two computational techniques. The first is a rigid body docking technique (Norel et al. 1994b, Helmer-Citterich and Tramontano 1994; Norel et al. 1998, 1999a, b); the second allows conformational flexibility of molecular parts, through hinge-bending motions (Sandak et al. 1995, 1996a, b, 1998, 1999). We note the geometrical representation of the molecular surface they currently employ (Connolly 1986; Norel et al. 1994b; Connolly 1983a, b; Lin et al. 1994, 1996) and pattern matching algorithms to detect geometric surface complementarity (Norel et al. 1994a; Fischer et al. 1995; Sandak et al. 1995). These techniques derive from computer vision and robotics (Wolfson 1991; Lamdan and Wolfson 1998; Lamdan et al. 1990). They are highly efficient, yielding candidate solutions in very short CPU matching times. They are straightforward to run, using an SGI workstation or a PC. However, despite all of these encouraging attributes, we are still faced with very serious difficulties, namely, the scoring and ranking of the obtained, predicted, docked configurations.

Are there then ways to quickly sift through potential docked configurations and rank the correct native configurations at the top of the obtained list? Analyses of protein-protein interfaces have illustrated that the binding interfaces do not necessarily have the largest extent of buried surface areas. Furthermore, native-like bound conformations do not manifest the largest nonpolar buried surface areas as compared to other potentially feasible docked solutions (Norel et al. 1999a). They do not contain the largest number of hydrogen bonds or the smallest number of unsatisfied buried polar groups. In solution, these are likely to be handled by surface motions, eliminating such unfavorable energy contributions. Hence, the problem is how, despite these hurdles, to still conceivably be able to detect candidate lead compounds/protein inhibitors. Here we outline one potential way of handling this problem, namely, through utilization of a *library of functional epitopes*. Such a library can be generated efficiently using techniques based on the same principles. We further consider the pros and cons of their utilization.

This chapter is divided into two parts. In the first, we describe computer vision based rigid and hinge-bending docking algorithms, and the generation of potential solutions. In the second, we focus on the generation of functional epitopes, and some attributes of binding epitopes that we have recently obtained (Hu et al. 2000).

Docking is an alternate, complementary tool to that of drug design. In drug design, the molecule is built, normally in the binding site of the receptor, by ligating groups of atoms, in a step-by-step, trial-and-error calculation. Here we focus on docking algorithms. In such algorithms the three-dimensional structures of the ligands are taken as single entities from the database.

19.2 Rigid-Body Docking

We have devised two rigid-body computer vision based docking techniques (Norel et al. 1994b, 1995; Fischer et al. 1995). In both the representation and the matching are 3D rotation and translation invariant. The principles of both algorithms are similar. Here we describe briefly only one. Additional details on this, and the second algorithm, are in Fischer et al. 1995.

To align the surfaces of two molecules in a complementary manner, we need to compute a rigid transformation that superimposes the surfaces without allowing one molecule to penetrate or overlap the other. To obtain hypotheses for such transformations it suffices to align a triplet of ordered non-collinear points (congruent triangles) from both molecules. However, it may happen that there are no three independent matching point-pairs between the receptor and the ligand. For docking, the points we utilize are those describing the molecular surfaces. These are computed to accurately represent the maxima (*holes*) and minima (*knobs*) of the shape function (Norel et al. 1994b, 1999a, b). We also compute the surface normal, associated with the point. Below, these points are dubbed *critical points*. The docking strategy utilizes only pairs of matching critical points with the additional geometric information on their normals. In order to compute a candidate rigid transformation, we need to detect a pair of critical points in both molecules that share the same internal distance, and, if superimposed, have opposing surface normals. This reduces the number of potential docked configurations, and concomitantly reduces the run-time complexity of the program.

For each pair of critical points from each of the molecules (any two critical points combination for the protein-protein docking; two holes for the receptor and two knobs for the ligand in the protein-drug cases) we compute a transformation invariant *signature*. The signature includes the distance between the two critical points (d); the two angles formed by the line between these critical points and their respective normals (α_1, α_2); and the torsion angle defined by the two normals and the line segment between them (ω). If the signatures of the ligand and of the receptor are compatible, the best rigid transformation (Schwartz and Sharir 1987) between the two pairs of the critical points is computed. In compatible pairs (1) a knob in one molecule matches a hole in the other; (2) the difference between the two distances (d) in the two molecules is less than a predefined threshold; and (3) the difference between the corresponding $\alpha's$ and the two $\omega's$ is also within the allowed limits. We impose stronger cumulative constraints on the alignment of the surface normals: the sums of the differences of the two $\alpha's$ and of the three angles should not exceed certain limits. In order to qualify as a candidate match, if one of the normals is not well aligned, the other must compensate and have a reasonable alignment. There are several advan-

tages in employing the surface normals in the signature, in addition to the critical points: (1) we need only two critical points in the interface area; (2) the combinatorics of finding correct matches is lower (we use pairs of points, rather than triplets or quartets, as Connolly has used in his original 1986 implemention); (3) the orientation of the normal is used for fast rejection of a large number of wrong solutions.

Because the matching is computed for local patches of the surface, it is essential to verify that the solution is viable for the entire protein. A problem may arise when the entire ligand molecule is brought to dock onto the entire receptor. It is conceivable that although there is a good complementarity at the matching interface, there could be an overlap between the two molecules elsewhere. To verify that the docked configurations do not seriously interpenetrate each other, we compute a scoring function. The function is based on geometric features: surface contact is awarded, and overlaps where ligand atom centers invade the outer shell of the molecular representation of the receptor are penalized, but retained. However, potential solutions where ligand atoms fall into the "core" of the receptor are rejected. By allowing a certain extent of intermolecular penetrations, we implicitly take into account molecular surface variability. The details of the scoring and ranking function along with a simple hydrophobicity function utilized in the docking of protein-protein pairs are described elsewhere (Norel et al. 1999a).

Inspection of the obtained complexed conformations immediately reveals that many molecular associations are relatively similar to each other. These represent virtually the same docked solution (Fischer et al. 1995). Clustering similar solutions both reduces the number of solutions and allows focusing on alternate configurations. A good clustering scheme should group similar solutions. However, at the same time, it should properly distinguish between alternate binding modes. The relative rotation between two conformations is computed from their individual rotations against the initial conformation (Norel et al. 1999a).

This computer vision based docking algorithm is highly efficient. Its fast CPU matching times—on the order of minutes on a PC, even for large protein-protein cases—allows large-scale docking trials of large and small molecules. We have carried out such experiments on about 230 receptor-ligand molecule pairs, including 26 protein-protein "bound" and 19 "unbound" protein-protein cases, where the structures have been determined separately (Norel et al. 1999a). These have variable, imperfectly matching surfaces. We have further docked about 160 cases of protein-drugs, 13 cases of protein-DNA, and 14 cases of DNA-drugs (Norel et al. 1999b). The molecules vary substantially both in size and in the hydrophobic/polar chemical nature of their surfaces. The location of the active site, or the identity of particular residues/atoms in either the receptor or the ligand, which participate in the binding,

has not been taken into account in any of these. As can be seen in Norel et al. (1999a, b), the quality of the results (low rmsds of the docked configuration as compared to the crystal-complex) and the speed of our techniques are highly attractive. Thus, based on this examination of the docking and the results it has obtained, we conclude that shape complementarity almost always enables attaining correct docked configurations. Nevertheless, it is insufficient for ranking.

Above, we have outlined a computer vision based algorithm for *rigid* docking. On the technical side, rigid docking algorithms are faster and obtain fewer candidate solutions than those allowing molecular flexibility. If the flexibility is limited, treating the molecules as rigid bodies will still detect the native bound conformations. Such cases are handled simply by the "error thresholds."

Below we extend the docking repertoire, by describing an efficient method for handling a special kind of flexibility, namely, allowing *hinge-bending* motions.

19.3 Hinge-Bending Flexible Matching

Here we dock a ligand onto a receptor surface, allowing hinge-bending movements of domains, subdomains, or any structural parts. The advantage of our algorithm is that we allow all angular rotations, although we still avoid a conformational space search (Sandak et al. 1995, 1996a, b, 1998, 1999). We pick a hinge point, to divide the molecule into two parts. However, we do not dock each of the molecular parts separately, necessitating a subsequent reconstruction of the consistently docked molecules. Instead, we dock all parts simultaneously. In particular, we utilize the position of the hinge from the start. Like pliers closing on a screw, in an automated fashion, the receptor closes on its ligand. Movements are allowed either in the ligand or in the larger receptor. Hence, the algorithm mimicks the so-called "induced" molecular fit. More than one hinge can be allowed in the docking. Interestingly, contrary to intuitive expectation, there is no increase in the *matching* (docking) times as the number of hinges increases. However, the ranking (scoring) times increase. This is the outcome of the necessity to examine two types of penetrations, both intramolecular part penetration and intermolecular as in rigid-body docking. The hinge-bending docking is illustrated in the schematic diagram of figure 19.1.

Hinge-bending movements are frequently related to molecular association (Gerstein et al. 1994). Such motions can involve domains, subdomains, loops, secondary structure elements, or be between any portions of the molecules connected by flexible joints. Currently, we have implemented the hinges at points and at bonds. Clearly, in reality, the only molecular movements that are possible are rotations around a covalent

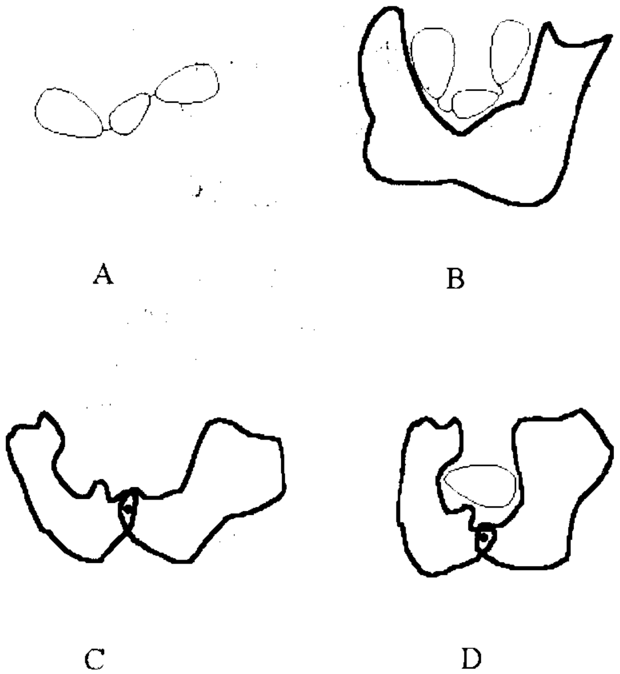


Figure 19.1
A schematic illustration of the molecular hinges allowed in ligands and receptors in the hinge-bending docking. (A) The hinges in the ligands. (B) The ligand is hinge-bent to optimally fit in the receptor active site. (C) The hinge is in the receptor. (D) The receptor is hinge-bent, closing on its ligand.

bond. Hence, in principle, we could have limited the matching to the simpler, one degree of freedom rotation. Nevertheless, by enabling full three-dimensional rotations around a *point*, rather than around a bond, we can implicitly take into account several rotations about consecutive or nearby bonds. If we so wish, the point rotation can be restricted to bond rotation. We can also permit the molecule to rotate around several, simultaneous hinges. In enabling several hinge motions to occur *simultaneously*, in effect we are simulating the cumulative effect of multiple mutations, each introducing a limited motion. In practice, to date, the algorithm has been implemented to enable two simultaneous hinges (Sandak et al. 1995, 1996a, b, 1998, 1999), although initial prototypes for multiple hinges are already being tested.

We have successfully applied our algorithm to a number of *bound* and *unbound* molecular configurations, achieving fast recognition times of their surfaces. As in the rigid-body case, the atomic coordinates are extracted from the PDB (Bernstein et al. 1997). The location of the hinge has been pre-defined, through a comparison of similar structures in different, that is, "open" and "closed" conformations, in cases where both types of structures exist. When we applied our method to molecules taken from bound conformations, we were able to reproduce the molecular association, in ex-

cellent agreement with the experimental crystal structure. Because the ligand and the receptor have been picked in their complexed form, near-native geometrical solutions correspond to those with near zero rotations and translations. The binding modes our algorithm has obtained have small rmsds compared to the native crystal complexes. The average rmsd of a correct solution is 1.4 Å, with average run-time for each complex around 1 min (on a SGI-Challenge R8000 machine). Additionally, for the bound cases, "correct" bound configurations typically rank high. In addition, geometrically well-fitting, alternate binding modes have also been generated (Sandak et al. 1995, 1996a, b, 1998, 1999).

19.4 Hinge-Bending Flexible Matching: The Algorithm

For simplicity and clarity, here we outline the description of this robotics-based method (Wolfson 1991) for the single hinge case, where the hinge is positioned in the ligand molecule and the receptor is assumed to be rigid. Nevertheless, although we describe the algorithm for flexible *ligands*, the roles of the ligands and the receptors are interchangeable. That is, the hinges can be positioned in the receptors as well. Elsewhere we describe further enhancements that we have implemented for the multiple hinges case (Sandak et al. 1999).

The algorithm is based on a *voting scheme* designed to find the most suitable ligands (out of a library of ligands), and for the respective transformations for matching each of their parts to a given receptor. Both the ligand and the receptor molecular surfaces are described by their 3D sets of "interest points." This molecular surface representation is essentially similar to the one described above for the rigid body docking. Details are given in Lin et al. refs. 36, 37. Here we proceed to outline the hinge-bending algorithm and its two phases, the *preprocessing* and the *recognition*. (For additional details of the algorithm, see Sandak et al. 1995, 1996a, b, 1998, 1999.)

An Overview of the Algorithm The ligands may undergo translations and rotations of their parts in order to optimally dock to the surface of the receptor. The ligand information is stored in a look-up table, which is invariant to this type of transformation. The table is generated in the *preprocessing* phase of the hinge-bending algorithm. The position of the hinge in each of the ligands is pre-determined by considerations of its more flexible joints. Nevertheless, because this robotics-based algorithm is fast, numerous other, alternate, hinge locations may be tried. The structure of the receptor is presented to the system in the second, *recognition* phase of the algorithm. If a ligand has an interest point configuration (i.e., a portion of the molecular surface) similar to the receptor interest point configuration, the algorithm

scores a match. This is done by casting a vote for this ligand, together with the computed location of its hinge. This hinge location is computed from the transformation between the corresponding receptor and ligand interest point surface configurations. The highest scoring (voted for) hinge locations of candidate ligands are sought. No knowledge of the binding site, or of the hinge locations relative to the receptor, is assumed.

In the *preprocessing* step, the ligand molecule (model) is described as a set of interest points. The (predetermined) hinge location is positioned at the origin of a 3D Cartesian coordinate frame. This frame is the *ligand frame*. The orientation of this frame is set arbitrarily. For each non-collinear triplet of interest points, in each of the ligand's parts, a unique triplet-based Cartesian frame is defined. This is the *triplet frame*. One way in which this can be done is by defining the origin at the first triplet point, the x-axis in the direction of the vector from the first point to the second, the z-axis, which is the normal to the triangle plane in the direction of the cross-product of the two vectors originating from the first triangle point, and the y-axis in the direction of the cross product of the x and z unit vectors. However, in practice, in order not to overlook any potential solution, we define three such frames, one for each triplet point. The *shape signature* of each triplet of points is the ordered triangle side lengths. This geometric shape signature of the triplet constitutes an address to a look-up hash table. The information that is stored at this entry at this address is the ligand identification, the part number, and the transformations between the *triplet frames* and the *ligand frame*.

In the *recognition* step, the molecular surface of the receptor is similarly described by its set of interest points. All non-collinear triplets of the interest points of the receptor are considered in the docking stage. For each of these triplets, the triplet-based Cartesian frames are computed. Each is the *receptor triplet frame*. The mode of calculation is as above. The lengths of the triangle sides of each of the triplets are similarly calculated. This calculation is invariant under rotation and translation. Thus (almost) congruent ligand triangles will result in similar values. The look-up table calculated in the preprocessing phase is entered using as an address of the currently computed ordered triplet of triangle side lengths of the receptor. For each ligand-record present at that entry in the table, a *candidate ligand frame* is computed, calculated by applying the pre-recorded ligand transformation at that (hash table) entry to the current *receptor triplet frame*. The origin of the candidate ligand frame is the candidate *hinge location*. We vote for the identity of the ligand molecule, along with the location and orientation of the *candidate, hinge-centered, ligand frame*. At the end of this procedure, we seek high scoring pairs of *(ligand, hinge location)*. This hinge location defines the 3D translation that the ligand would need to undergo in this

candidate docking. The appropriate rotations are calculated separately for each part only at a later scoring and filtering stage. In the current step, hinges that have received a large number of votes are selected.

This hinge-bending algorithm exploits the fact that both parts of the molecule share the same hinge. The essential point here is that the way it is taken into account is by locating the origin of the reference frame of the ligand *at the hinge*. In this way, both parts contribute votes to a reference frame at the same location, although the orientations of both parts with respect to each other may differ. Most importantly, by picking up votes from both molecular parts, a ligand, which might otherwise have only a small portion of surface complementary to the receptor surface in each of the parts, may still score high. Thus, although each of the individual parts of the ligand can obtain an insignificant score, the sum of the votes obtained from both of the ligand's parts may yield an overall acceptable match, which can be automatically detected.

This algorithm is general. It can handle the rigid docking as a particular case. For the rigid-body docking, the ligand reference frame is located arbitrarily.

Above, we have outlined the algorithm for the case of a single hinge. However, it may be extended to multiple hinges. There, rather than pick a single ligand frame, multiple ligand frames are defined. Each of these frames is centered at a different, pre-defined hinge. Next, in the preprocessing step, for each ligand triplet in a single part, we encode the transformations of its triplet frame to all ligand frames of that part. Hence, if a part has two hinges (as might be the case if the part is the middle of a protein), we will store two transformations for each triplet. On the other hand, if a part has only one hinge (i.e., if it is an edge part), we store only one transformation. (A drug-ligand with a branched structure may have more than two hinges per part). The recognition phase is unchanged. The exception is that each receptor triplet participates in the voting for as many frames as the number of different transformations stored in its table entry. Run times at the different steps of the algorithm on several examples are given by Sandak et al. (1999).

19.5 The Ranking Problem

Above we have described two algorithms for docking ligands to the surfaces of their respective receptors. The two methods are very efficient, producing high-quality results for cases where the complementarity between the surfaces of the molecules is relatively good. Nevertheless, despite the advantages of these techniques, they face considerable difficulties as the fit deteriorates. This problem is uniformly encountered by all current

methodologies. If docking initiates from entire molecular surfaces where the structures of the molecules have been determined separately, that is, when they are in their uncomplexed conformation, a range of solutions is obtained. The problem of how to rank the solutions, with near-native solutions ranking near the top of the list, is still a major hurdle. Because the number of candidate solutions may be large, detailed chemical calculations are unrealistic, and fast empirical approaches have yielded unsatisfactory results. Among the many schemes that have been tested over the last years we find derivation and utilization of pairwise, knowledge-based potentials, either residue-based or atom-based; calculations of the nonpolar buried surface area, or of the total buried surface area; inspection and counts of hydrogen bonds and salt bridges; minimization of the number of buried polar groups; and minimization of the energies. Any of these may be satisfactory for some cases; however, they fail for the others. Thus, even if the docking programs are good and can scan quickly large databases of drugs or inhibitors, the ranking problem still constitutes a huge stumbling block.

19.6 Binding Epitopes

The definitions of binding epitopes vary. In general, the term refers to a recurring pattern of molecular surface in a family (or, families) of proteins, where at least in one family member the site is known to be a binding site. A similar binding epitope can also be observed across family boundaries. To construct a library of binding epitopes we may start by picking a known site as a *pattern*, and search for similar portions of surface, or of arrangement of residues at/near the surface, in other molecules.

19.6.1 Characteristics of Binding Sites

Locating a priori unknown active (binding) sites on the surfaces of enzymes or receptors is important for a number of reasons. First, detection of binding sites is the initial step in the design of drugs. Second, being able to identify active sites on the surfaces of proteins enables their modification to enhance their activity toward specific ligands and functions. Knowledge of likely binding sites enables focusing on these locations in the docking simulations. Finally, they can further be used to filter and rank the obtained solutions at later stages. Although the former is more efficient, we may nevertheless also wish to obtain alternate geometrically feasible configurations.

In an insightful review, Ringe (1995) surveys the definition of "what makes a binding site a binding site." She describes some guidelines, and proposes that binding sites are in general depressions in the protein surface, "in which there is greater than

average degree of exposure of hydrophobic groups." These frequently contain disordered, easily displaced water molecules. Ringe further suggests that conformationally flexible residues at the binding site may be particularly useful in replacing the disordered water by the competing ligand. Laskowski et al. (1996) have also examined the active sites. Their extensive study indicates that active sites of enzymes typically consist of large clefts. Their study shows that in the majority of single chain enzymes, the ligand binds in the largest cleft. Hence, active sites of enzymes may well be identified using geometrical criteria alone. Peters et al. (1996) utilized alpha-shapes (Edelsbrunner and Mucke 1994), a computational geometry tool, in an automated search for ligand binding sites on protein surfaces. This strictly geometry-based algorithm has also found a correlation between the depth of the clefts and enzyme active sites. In addition, a correlation between patches of hydrophobic surfaces and binding sites has also been noted (e.g., Young et al. 1994; Vakser et al. 1994; Clackson and Wells 1995). Both the results of Peters et al. (1996), and of Laskowski et al. (1996), confirm an older observation (Connolly 1986), that protein-protein binding sites are characteristically more shallow, unlike the enzyme active sites.

A straightforward approach to locate an active site in proteins if their structures are available is to select a known active site of a (model) protein and use it as a template, searching for similar ones in target proteins. This can be done by initially describing the molecular surfaces of the proteins and picking the points that faithfully represent the surfaces. Alternatively, we can use the location of the atoms that line the surface, or are near it. In the latter case, we may either select the coordinates of surface atoms or of C_α-atoms. C_α-atoms may be those whose atoms line the molecular surface, or more likely, they will be in a certain shell from the surface, and will belong to functionally important residues, such as those of the catalytic triad of the serine proteases. Such search methods need to be flexible enough to allow both inexact and partial matching of the model and target molecules.

Molecular surfaces are flexible. In particular, that may hold for active site regions. Flexibility may be particularly advantageous for proteins with a broad range of ligands/substrates. The less specific the receptor-ligand (enzyme-substrate) interactions, and the broader the range of binding, the more flexible the binding site is likely to be. Further, enzymes and proteins having a broad range of bound substrates/ligands may also make substantial use of water molecules to mediate binding to their ligands (e.g., Ringe 1995; Bhat et al. 1994; Ladbury 1996). Ladbury (1996) outlines the rationale and the consistent evidence, arguing that water is likely to have a dual role in binding. Water molecules may increase the promiscuity of the binding; however, water might also increase the specificity and affinity. In general, bridging water molecules are not accounted for in molecular surface matching algorithms.

That is largely due to the difficulty in predicting their locations on the surfaces of the proteins. A further difficulty is to pinpoint the residues that are critical for binding. Residues may be observed to be in contact with their cognate ligands in a number of different members of the family. This, however, does not necessarily imply that these residues are critical for binding. There is a growing body of examples showing that mutating such residues still enables molecular association. What happens is that owing to side chain flexibility, side chains that are nearby move, taking over the role played by the residue that was originally at that location. A binding epitope should therefore always be taken with a grain of salt. Still, as initial guesses, especially when a certain sloppiness in the stipulated requirements is allowed, they are useful to have.

19.6.2 Detection of Conserved Residues at Binding Sites

A method such as Geometric Hashing (Nussinov and Wolfson 1991; Tsai et al. 1996a, b) enables carrying out a comprehensive and systematic structural analysis of protein-protein interfaces. Because the crystal structures of the families of these interfaces are geometrically similar (Tsai et al. 1996b), the dataset can be used to study structural characteristics of protein-protein interfaces. In particular, such a dataset may be used to address questions such as, given a geometric similarity among members of a protein-protein interface family, how similar are their binding surfaces? Which residues at the interface are conserved? What are the determinants of the binding (hydrophobicity, electrostatic, etc.)? Which residues are preferred, avoided, or neutral at the interface? Which residues are poor interface formers? These questions have important consequences for both protein-protein binding prediction and protein ligand design. An insight into such questions should enable searches for homologous, potential sites in other structures, where the existence/location of these sites are unknown.

A number of studies addressed the question of which are the critical residues at protein binding sites (Clackson and Wells 1995). The studies examined either a single or a few protein-protein interfaces. Bogan and Thorn (1998) carried out an extensive analysis of alanine scanning mutagenesis. However, although the total number of mutations was large, the number of protein interfaces was small, with some of the interfaces closely related.

Recently, we have shown (Hu et al. 2000) that although binding sites are hydrophobic, they are seeded with conserved polar residues at specific locations, possibly serving as energy "hot spots." Our results have confirmed and generalized the alanine scanning data analysis (Bogan and Thorn 1998). In that earlier study, Trp, Arg, and Tyr were observed to constitute energetic hot spots. They were rationalized by their polar interactions and by their surrounding rings of hydrophobic residues. Never-

theless, there was no compelling reason to explain why it is specifically these residues. Our study has illustrated that other polar residues are similarly conserved. Conserved residues that are in contact across the protein-protein interfaces have been observed in all the examined families. These results are based on 11 clustered interface families, comprising a total of 97 crystal structures. The families have at least five members, with sequence similarity between the members in the range of 20–90 percent. Because the matching was carried out by the Geometric Hashing structural comparison algorithm (Nussinov and Wolfson 1991; Tsai et al. 1996a, b), the conserved residues are at spatially similar environments.

Additionally, for the enzyme-inhibitors, we have observed that residues are more conserved at the interfaces than at other locations. On the other hand, antibody-protein interfaces have similar surface conservation as compared to their corresponding linear sequence alignment, consistent with the suggestion that evolution has optimized protein interfaces for function.

Protein-protein interfaces are largely hydrophobic, with hydrophobicity being a dominant force in protein-protein interactions (Young et al. 1994; Korn and Burnett 1991). Analysis of protein-protein interfaces has shown that the extent of hydrophobicity, as measured by the nonpolar buried surface area between the two chains, may vary to a large extent between the interfaces (Tsai et al. 1997). Consistently, there are also indications of the existence of hydrophilic regions (Korn and Burnett 1991; Janin et al. 1988). Two recent studies have highlighted the importance of hydrophilic interactions in biological function (Tormo et al. 1999; Wang et al. 1999; Xu et al. 1997). With regard to specific residues, the reported preferences vary from case to case, mainly due to the change in the systems studied. In studies of a specific system, it is difficult to differentiate between residue conservation conferring binding specificity and conservation owing to the role of the residues in constituting energy hot spots.

19.6.3 Distribution of Interface Residues

We have examined the distributions of all interface residues, of identically matched residues between the representative of the family and each of the family members, and of conserved residues in the whole family. For individual families, there are significant preferences of certain residues to be at the interfaces. However, when examining the counts of identically matched residues, the preferences become much less significant.

Although the distribution of all interface residues yields simple statistics, the propensities of conserved residues provide clues to the more important driving forces (Hu et al. 2000). Table 19.1 summarizes the propensities in terms of hydrophobic or

Table 19.1
Propensities in terms of hydrophobic or hydrophilic residue classifications for all residues, identical residues, and conserved residues

Residue classifications[a]	Total interface residures	Identically matched interface residues	Conserved interface residue
Hydrophobic	5.9	4.6	5.7
Aromatic	6.5	3.7	5.4
Hydrophobic + Aromatic	12.4	8.3	11.1
Polar	9.6	7.0	14.9

[a] Hydrophobic residues include Ala, Vla, Pro, Met, Ile, and Leu; polar residues include Asp, Glu, Lys, Arg, Ser, Thr, Asn, and Gln; aromatic residues include Phe, Tyr, His, and Trp.

hydrophilic residue classifications. Here aromatic residues are considered hydrophobic. The table shows that with respect to the total number of interface residues, hydrophobic residues have higher propensities than polar residues (12.4 : 9.6), consistent with previous, statistically based studies (Young et al. 1994; Tsai et al. 1997). However, remarkably, the preference of hydrophobic residues decreases when analyzed in terms of the propensities of identically matched residues (8.3 : 7.0). In terms of conserved residues, polar residues are observed to dominate (11.2 : 14.9). Hence, we observe a trend of preferred conservation of polar residues in protein-protein interfaces.

A comparison of our conserved interface residues with experimentally identified energetical hot spots illustrates that the two most conserved residues are cysteine and arginine (figure 19.2). The high propensity of cysteine stems from the disulfide bond conservation, either to maintain the interface structure or to chemically bind interfaces together. The identification of cysteine validates our conserved residue propensity computations.

Table 19.2 and figure 19.3 show that our selection of conserved residues corresponds to experimentally identified hot spots. There is a surprising match between the propensities of most of our conserved residues and the residue enrichment in hot spots compiled from the database of alanine scanning mutagenesis. Except for a few outliers, the correlation coefficient of the experimentally determined amino acid enrichment and our computed conservation propensity is 0.72 (Bogan and Thorn 1998). Of the unmatched residues, alanine is clearly incomparable. Cysteine is favored in our database owing to its forming covalent bonds. Tryptophan has a lower propensity in our analysis, probably because of its rareness. Serious disagreement is found for lysine. Lysine is well represented in our interface database. However, although in our analysis we failed to identify conserved lysine residues, lysine has a good enrichment in the alanine scanning data. A possible reason for this failure may relate to our

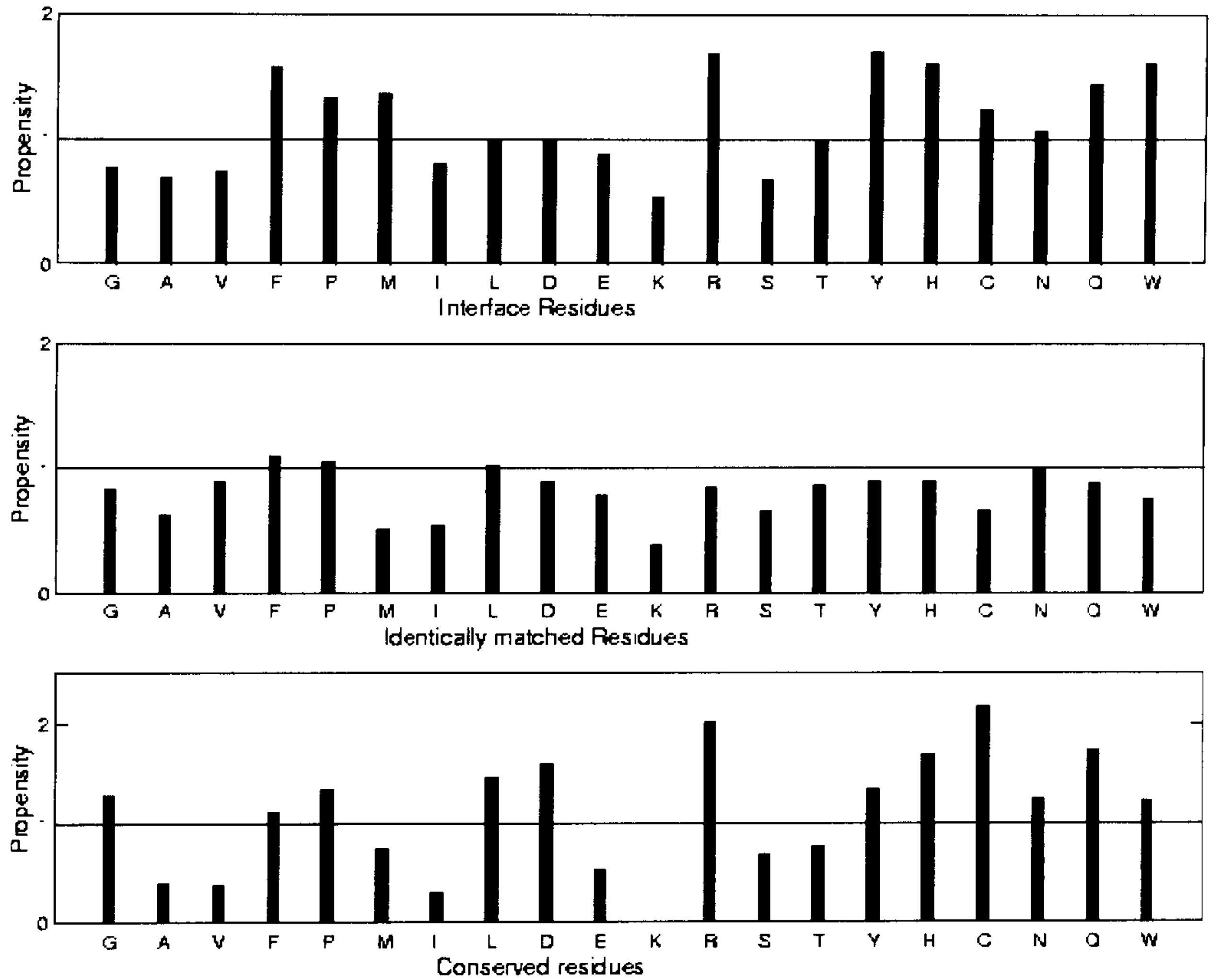


Figure 19.2
Propensities of all interface residues (A), of identically matched residues (B) and of conserved residues (C).

high criteria of cutoff percentage (80 percent) for a conserved interface residue. Alternatively, lysine is a highly flexible surface residue, projecting into the solvent. It is quite possible that it is not detected as a matched residue pair in the superpositioning owing to slightly larger shifts in its position. At first sight, it appears that there is also a serious disagreement between our results and the alanine scanning with respect to leucine. However, if we combine the contributions of leucine and isoleucine, we obtain a good agreement (1.80 from alanine scanning, 1.74 from conservation propensity). This agreement further addresses Bogan and Thorn's question as to why isoleucine has a higher frequency than leucine in their database. Taken together, the solution may be that one should consider the sum of leucine and iso-

Table 19.2
Conserved residue propensities vs. experimentally determined enrichment in hot spots

Residue	Propensity to conservation[a]	Enrichment in hot spots[b]
Val	0.36	0
Ser	0.67	0.21
Thr	0.75	0.28
Gly	1.29	0.45
Met	0.74	0.54
Phe	1.12	0.56
Gln	1.73	0.58
Glu	0.52	0.68
Asn	1.24	0.93
Lys	0	1.17
Pro	1.33	1.25
His	1.68	1.49
Asp	1.59	1.67
Ile	0.29	1.79
Leu	1.45	0.01
Leu + Ile	1.74	1.80
Tyr	1.35	2.29
Arg	2.01	2.47
Trp	1.22	3.91

[a]The propensity of conservation (P_k) of a residue to occur at the interface is calculated as the fraction of the count of residue k in the interface as compared with its fraction in the whole chains. $P_k = (n_k/n)/(N_k/N)$, where n_k is the number of conserved residues of type k at the interface, n is the number of residues at the interface, N_k is the number of residues of type k in the chain, and N is the total number of residues in the chain.
[b]Enrichment in hot spot gives fold enrichment of that residue type in hot spots ($\Delta\Delta G \geq 2$ kcal/mol in alanine mutations) over whole database of 2,325 alanine mutations (Tsai et al. 1997).

leucine, as we did in figure 19.3. There are also disagreements in the lower third of the table, particularly in Gln, Phe, and Gly. Although the difference in Gly can be related to flexibility, we have no explanation for the more frequent occurrence in Gln and Phe in our interface families as compared to the alanine scanning, apart from the limited number of dissimilar interfaces examined by the alanine scanning (Hu et al. 2000). In general, we see a higher preponderance in conserved polar residues (His, Asn, Gln, Thr, Ser), or partially polar (Phe, Met), as compared to the alanine scanning. Further, Bogan and Thorn (1998) have examined the location of the hot spots across interfaces. They found that the hot spots are usually located around the center of the interfaces, and hence protected from bulk solvent. This is also observed in our location of conserved interface residues. We note, however, that the number of datapoints, particularly the number of conserved interface residues, is small. Some of the

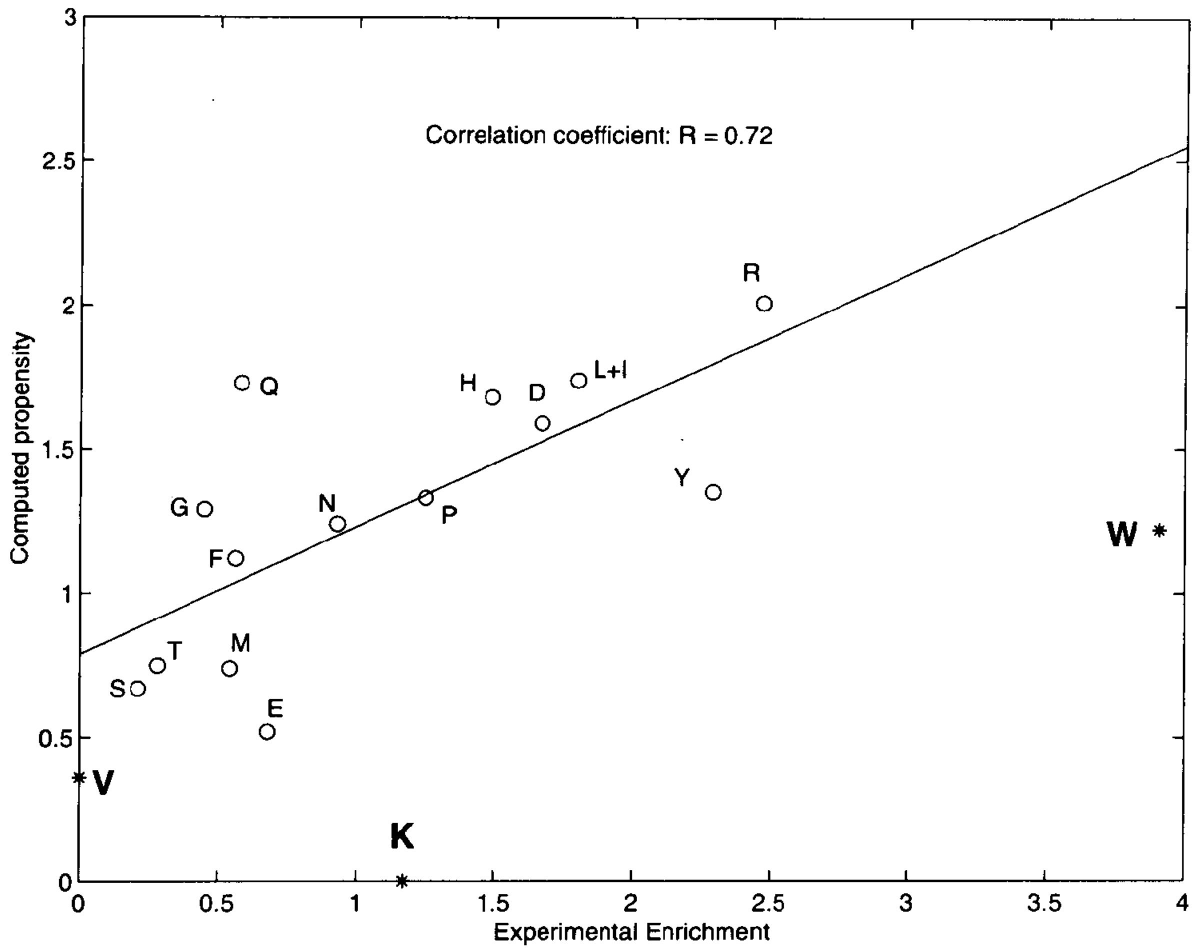


Figure 19.3
A correlation of experimentally determined amino acid enrichment and our computed conservation propensity. Residues are indicated by their one-letter code. Outliers are indicated. They are not included in the least square fitting. Note that Val is classified as an outlier due to its zero value in the experimental results.

outliers may be due to poor statistics. Another point worthy of note is with regard to the residue conservation and protein function. Although these residues recur in all interfaces, with the interfaces belonging to different families and fulfilling different functions, it is still possible that some conservation is due to protein function.

Hence, although overall interfaces manifest a higher frequency of occurrence of hydrophobic residues (Young et al. 1994; Korn and Burnett 1991, Tsai et al. 1997), the polar residues are those that are preferentially conserved. Within these, Bogan and Thorn find a special enrichment of Trp, Arg, Tyr, Asp, Pro, and His. Analysis of the residue conservation in a larger number of interfaces shows Arg, Gln, His, Asp, Pro, and Asn to be especially enriched.

19.6.4 Binding Epitopes and Docking: Conclusions

Our goal was to detect binding epitopes shared by members of the same family. In particular, we sought to probe the potential existence of common principles that hold in general for protein-protein binding sites, and as such can be used in docking. Several observations have been made in the literature. First, binding sites are generally hydrophobic, indicating that the hydrophobic effect plays a major role in protein-protein interactions. Second, the extent of the hydrophobicity is variable, with some interfaces considerably more hydrophobic than others. Consistently, hydrophilic side chains play a more important role in binding than in folding. Third, for some interfaces, it has been shown that electrostatics plays a major role, steering the ligand onto the binding site of the receptor. Fourth, the availablility of a large number of clustered interfaces, combined with the Geometric Hashing structural comparison tool, which enables superimposing the interfaces despite the fact that they are composed of discontinuous pieces of the chains, provide powerful tools.

The interface families that we have examined show a preference for conservation of polar residues at their interfaces. In particular, conserved interface residues are strongly correlated with the experimentally identified "hot spots" compiled from the database of experimental alanine scanning mutagenesis. Specifically, the more highly conserved residues in our analysis are Arg, Gln, His, Asp, Pro, and Asn. Bogan and Thorn (1998) argue that energetic hot spots are critically important for the affinity of a protein interface. The fact that these residues tend to be conserved at specific locations indicates that they may constitute binding epitopes (functional epitopes). As such, they may be utilized in searches for potential unknown binding sites. They may further be engineered in binding site design. Conserved interface residues may play a dual role in binding, both thermodynamic and kinetic.

Additionally, the surface alignment identified conserved residues in the eleven clustered protein-protein interfaces. In enzyme-inhibitors, we find that the residues are more conserved at the interfaces than in other locations in the proteins, as shown by multiple sequence alignments. In contrast, antibody-protein antigen interfaces illustrate similar surface conservation as compared to linear sequence alignment, consistent with the proposition that evolution has optimized protein interfaces to achieve optimal function, as conserved residues are more likely to be optimized.

The main advantage in the utilization of binding epitopes in docking is that they are inherently resilient to the atomic conformational detail and provide a practical (albeit statistically based) way of treating surface variability and flexibility. They further enable the utilization of modeled structures, in addition to high-resolution ones.

Acknowledgments

We thank Drs. Chung-Jung Tsai, Sandeep Kumar, and in particular Dr. J. V. Maizel for discussions and for encouragement. The research of R. Nussinov and H. J. Wolfson in Israel has been supported in part by the Magnet grant, by the Ministry of Science grant, and by the "Center of Excellence in Geometric Computing and Its Applications," funded by the Israel Science Foundation (administered by the Israel Academy of Sciences). The research of H. J. W. is partially supported by the Hermann Minkowski-Minerva Center for Geometry at Tel Aviv University. This project has been funded in whole or in part with federal funds from the National Cancer Institute, National Institutes of Health, under contract number NO1-CO-56000. The content of this publication does not necessarily reflect the view or policies of the Department of Health and Human Services, nor does mention of trade names, commercial products, or organization imply endorsement by the U.S. government.

References

Bacon, D., and Moult, J. (1992). Docking by least-square fitting of molecular surface patterns. *J. Mol. Biol.* 225: 849–858.

Bernstein, F. C., Koetzle, T. F., Williams, G. J. B., Meyer, E. F., Brice. M. D., Rodgers, J. R., Kennard, O., Shimanouchi, T., and Tasumi, M. (1977). The protein data bank: A computer-based archival file for macromolecular structures. *J. Mol. Biol.* 112: 535–542.

Bhat, T. N., Bentley, G. A., Boulot, G., Greene, M. I., Tello, D., Dall'Acqua, W., Souchon, H., Schwarz, F. P., Mariuzza, R. A., and Poljak, R. J. (1994). Bound water molecules and conformational stabilization help mediate an antigen-antibody association. *Proc. Natl. Acad. Sci., USA* 91: 1089–1093.

Bogan, A. A., and Thorn, K. S. (1998). Anatomy of hot spots in protein interfaces. *J. Mol. Biol.* 280: 1–9.

Cherfils, J., Duquerroy, S., and Janin, J. (1991). Protein-protein recognition analyzed by docking simulations. *Proteins: Struct. Funct. Genet.* 11: 271–280.

Cherfils, J., and Janin, J. (1993). Protein docking algorithms: Simulating molecular recognition. *Curr. Opin. Struct. Biol.* 3: 265–269.

Clackson, T., and Wells, J. A. (1995). A hot spot of binding energy in a hormone-receptor interface. *Science* 267: 383–386.

Connolly, M. (1983a). Analytical molecular surface calculation. *J. Appl. Cryst.* 16: 548–558.

Connolly, M. (1983b). Solvent-accessible surfaces of proteins and nucleic acids. *Science* 221: 709–713.

Connolly, M. (1986). Shape complementarity at the hemoglobin $\alpha_1\beta_1$ subunit interface. *Biopolymers* 25: 1229–1247.

Edelsbrunner, H., and Mucke, E. P. (1994). Three-dimensional alpha-shapes. *ACM Transact. Graph.* 13: 43–72.

Fischer, D., Lin, S. L., Wolfson, H., and Nussinov, R. (1995). A geometry-based suite of molecular docking processes. *J. Mol. Biol.* 248: 459–477.

Gabb, H. A., Jackson, R. M., and Sternberg, M. J. E. (1997). Modeling protein docking using shape complementarity, electrostatics and biochemical information. *J. Mol. Biol.* 272: 106–120.

Gerstein, M., Lesk, A. M., and Chothia, C. (1994). Structural mechanism for domain movements in proteins. *Biochemistry* 6739–6749.

Goodsell, D., and Olson, A. (1990). Automated docking of substrates to proteins by simulated annealing. *Proteins: Struct. Funct. Genet.* 8: 195–202.

Helmer-Citterich, M., and Tramonato, A. (1994). Puzzle: A new method for automated protein docking based on surface shape complementarity. *J. Mol. Biol.* 235: 1021–1031.

Hu, Z., Ma, B., Wolfson, W., and Nussinov, R. (2000). Conservation of polar residues as hot spots at protein-protein interfaces. *Proteins* 39: 331–342.

Janin, J., Miller, S., and Chothia, C. (1988). Surface, subunit interfaces and interior of oligomeric proteins. *J. Mol. Biol.* 204: 155–164.

Jiang, F., and Kim, S. (1991). Soft docking: Matching of molecular surface cubes. *J. Mol. Biol.* 219: 79–102.

Jones, G., Willet, P., Glen, R., Leach, A., and Taylor, R. (1997). Development and validation of a genetic algorithm for flexible docking. *J. Mol. Biol.* 267: 727–748.

Kasinos, N., Lilley, G., Subbarao, N., and Haneef, I. (1992). A robust and efficient automated docking algorithm for molecular recognition. *Prot. Engng.* 5: 69–75.

Katchalski-Katzir, E., Shariv, I., Eisenstein, M., Friesem, A., Aflalo, C., and Vakser, I. (1992). Molecular surface recognition: Determination of geometric fit between protein and their ligands by correlation techniques. *Proc. Natl. Acad. Sci. USA* 89: 2195–2199.

Korn, A. P., and Burnett, R. M. (1991). Distribution and complementarity of hydropathy in multisubunit proteins. *Proteins* 9: 37–55.

Kuntz, I., Blaney, J., Oatley, S., Langridge, R., and Ferrin, T. (1982). A geometric approach to macromolecule-ligand interactions. *J. Mol. Biol.* 161: 269–288.

Ladbury, J. E. (1996). Just add water! The effect of water on the specificity of protein-ligand binding sites and its potential application to drug design. *Chemistry & Biology* 3: 973–980.

Lamdan, Y., and Wolfson, H. J. (1988). Geometric hashing: A general and efficient model-based recognition scheme. In *Proc. 2nd Int. Conf. on Computer Vision*, Bajcy R. and Ullman S., eds., 238–249. IEEE Comp. Soc. Press.

Lamdan, Y., Schwartz, J. T., and Wolfson, H. J. (1990). Affine invariant model-based object recognition. *Trans. Robotics and Autom.* 6(5): 578–589.

Laskowski, R. A., Luscombe, N. M., Swindells, M. B., and Thornton, J. M. (1996). Protein clefts in molecular recognition and function. *Prot. Sci.* 5: 2438–2452.

Lengauer, T., and Rarey, M. (1996). Computational methods for biomolecular docking. *Curr. Opin. Struct. Biol.* 6: 402–406.

Lin, S. L., Nussinov, R., Fischer, D., and Wolfson, H. J. (1994). Molecular surface representation by sparse critical points. *Proteins: Struct. Funct. Genet.* 18: 94–101.

Lin, S. L., and Nussinov, R. (1996). Molecular recognition via face center representation of a molecular surface. *J. Mol. Graphics.* 14: 78–90.

Norel, R., Fischer, D., Wolfson, H., and Nussinov, R. (1994a). Molecular surface recognition by a computer vision based technique. *Protein Engineering* 7: 39–46.

Norel, R., Lin, S. L., Wolfson, H., and Nussinov, R. (1994b). Shape complementarity at protein-protein interfaces. *Biopolymers* 34: 933–940.

Norel, R., Lin, S. L., Wolfson, H., and Nussinov, R. (1995). Molecular surface complementarity at protein-protein interfaces: The critical role played by surface normals at well placed, sparse points in docking. *J. Mol. Biol.* 252: 263–273.

Norel, R., Lin, S. L., Xu, D., Wolfson, H., and Nussinov, R. (1998). Molecular surface variability and induced conformational changes upon protein-protein association. In *Structure, Motion, Interaction and*

Expression of Biological Macromolecules, Sarma, R. H. and Sarma, M. H., eds., 33–51. Albany, N.Y., Adenine Press.

Norel, R., Petrey, D., Wolfson, H., and Nussinov, R. (1999a). Examination of shape complementarity in docking of *unbound* proteins. *Proteins: Struct. Funct. Genet.* 36: 307–317.

Norel, R., Wolfson, H., and Nussinov, R. (1999b). Small ligand recognition: Solid angles surface representation and shape complementarity. *Combinatorial Chemistry & High Throughput Screening* 2: 177–191.

Nussinov, R., and Wolfson, H. J. (1991). Efficient detection of motifs in biological macromolecules by computer vision techniques. *Proc. Natl. Acad. Sci USA* 88: 10495–10499.

Nussinov, R., and Wolfson, H. (1999a). Efficient computational algorithms for docking, and for generating and matching a library of functional epitopes. I. Rigid and flexible hinge-bending docking algorithms. *Combinatorial Chemistry & High Throughput Screening* 2: 249–259.

Nussinov, R., and Wolfson, H. (1999b). Efficient computational algorithms for docking, and for generating and matching a library of functional epitopes. II. Computer vision-based techniques for the generation and utilization of functional epitopes. Rigid and flexible hinge-bending docking algorithms. *Comb. Chem. High Throughput Screen.* 2: 261–269.

Peters, K. P., Fauck, J., and Frommel, C. (1996). The automatic search for ligand binding sites in proteins in proteins of known three-dimensional structure using only geometric criteria. *J. Mol. Biol.* 256: 201–213.

Rarey, M., Wefing, S., and Lengauer, T. (1996). Placement of medium-sized molecular fragments into active sites of proteins. *J. Comp.-Aided Mol. Design* 10: 41–54.

Ringe, D. (1995). What makes a binding site a binding site? *Curr. Opin. Strct. Biol.* 5: 825–829.

Sandak, B., Nussinov, R., and Wolfson, H. J. (1995). An automated computer-vision & robotics based technique for 3D flexible biomolecular docking and matching. *Comp. Appl. BioSci* 11: 87–99.

Sandak, B., Nussinov, R., and Wolfson, H. J. (1996a). *Docking of Conformationally Flexible Proteins. Seventh Symposium on Combinatorial Pattern Matching.* Laguna Beach, CA: Springer Verlag.

Sandak, B., Wolfson, H. J., and Nussinov, R. (1996b). Hinge-bending at molecular interfaces: Automated docking of a dihydroxyethylene-containing inhibitor of the HIV-1 protease. *J. Biomol. Struct. & Dynamics*, Proceedings of the Ninth Conversation, Sarma, R. H. and Sarma, M. H., eds. New York: Adenine Press, 1: 233–252.

Sandak, B., Wolfson, H. J., and Nussinov, R. (1998). Flexible docking allowing induced fit in proteins: Insights from an open to closed conformational isomers. *Proteins: Struct. Funct. Genet.* 32: 159–174.

Sandak, B., Nussinov, R., and Wolfson, H. J. (1999). A method for biomolecular structural recognition and docking allowing conformational flexibility. *J. Comput. Biol.* 5: 631–654.

Schwartz, J. T., and Sharir, M. (1987). Identification of partially obscured objects in two-dimensions by matching of noisy "characteristic curves." *Int. J. Robotics Res.* 6(2): 29–44.

Shoichet, B., and Kuntz, I. (1991). Protein docking and complementarity. *J. Mol. Biol.* 221: 327–346.

Tormo, J., Natarajan, K., Margulies, D., and Mariuzza R. A. (1999). Crystal structure of a lectin-like natural killer cell receptor bound to MNC class I ligand *Nature* 402: 623–631.

Tsai, C. J., Kumar, S., Ma, B., and Nussinov, R. (1999). Folding funnels, binding funnels and protein function. *Protein Sci.* 8: 1181–1190.

Tsai, C.-J., Lin, S. L., Wolfson, H., and Nussinov, R. (1996a). Techniques for searching for structural similarities between protein cores, protein surfaces and between protein-protein interfaces. *Techniques in Protein Chem.* 7: 419–429.

Tsai, C.-J., Lin, S.-L., Wolfson, H., and Nussinov, R. (1996b). A dataset of protein-protein interfaces generated with a sequence-order-independent comparison technique. *J. Mol. Biol.* 260: 604–620.

Tsai, C. J., Lin, S. L., Wolfson, H. J., and Nussinov, R. (1997). Studies of protein-protein interfaces: A statistical analysis of the hydrophobic effect. *Protein Sci.* 6: 53–64.

Vakser, I. A., and Aflalo, C. (1994). Hydrophobic docking: A proposed enhancement to molecular recognition techniques. *Proteins: Struct. Funct. Genet.* 20: 320–329.

Wang, H. (1991). Grid-search molecular accessible algorithm for solving the protein docking problem. *J. Comp. Chem.* 12: 746–750.

Wang, J. H., Smolyar, A., Tan, K., Liu, J. H., Kim, M., Sun, Z. J., Wagner, G., and Reinherz, E. L. (1999). Structure of a heterophilic adhesion complex between the human CD2 and CD58(LFA-3) counterreceptors. *Cell* 97: 791–803.

Wallqvist, A., and Covell, D. (1996). Docking enzyme-inhibitor complexes using a preference-based free-energy surface. *Proteins: Struct. Funct. Genet.* 25: 403–419.

Wolfson, H. J. (1991). Generalizing the generalized Hough transform. *Pattern Recog. Lett.* 12: 565–573.

Xu, D., Lin, S. L., and Nussinov, R. (1997). Protein binding versus protein folding: The role of hydrophilic bridges in protein association. *J. Mol. Biol.* 265: 68–84.

Young, L., Jernigan, R. L., and Covell, D. G. (1994). A role for surface hydrophobicity in protein-protein recognition. *Protein Sci.* 3: 717–729.

Contributors

Hue Sun Chan
Associate Professor, Department of Biochemistry
Faculty of Medicine, University of Toronto
Toronto, Ontario, Canada

Alexander Diemand
GlaxoWellcome Experimental Research (GWER) and Scientific Computing (World-Wide)
Geneva, Switzerland

Nadia El-Mabrouk
Department of Information and Systems Reseach
University of Montreal
Montreal, Quebec, Canada

Susumu Goto
Institute for Chemical Research
Kyoto University
Kyoto, Japan

Nicolas Guex
Head and Director, GlaxoWellcome Experimental Research (GWER) and Scientific Computing (World-Wide)
Geneva, Switzerland

Xiaoqiu Huang
Associate Professor, Department of Computer Science
Iowa State University
Ames, Iowa

Tao Jiang
Professor, Department of Computer Science and Engineering
University of California, Riverside
Riverside, California

Minoru Kanehisa
Professor, Institute for Chemical Research
Kyoto University
Kyoto, Japan

Hüseyin Kaya
Department of Biochemistry
Faculty of Medicine, University of Toronto
Toronto, Ontario, Canada

Paul Kearney
Assistant Professor, Department of Computer Science
University of Waterloo
Waterloo, Ontario, Canada

Ming Li
Professor, Department of Computer Science
University of California, Santa Barbara
Santa Barbara, California

Jun S. Liu
Professor, Department of Statistics
Harvard University
Cambridge, Massachusetts

Buyong Ma
Laboratory of Experimental and Computational Biology
National Cancer Institute
Frederick, Maryland

Ruth Nussinov
Professor, Laboratory of Experimental and Computational Biology
National Cancer Institute
Frederick, Maryland

Manuel C. Peitsch
Head and Director, GlaxoWellcome Experimental Research (GWER) and Scientific Computing (World-Wide)
Geneva, Switzerland

David Sankoff
Professor, Center for Mathematical Research
University of Montreal
Montreal, Quebec, Canada

Torsten Schwede
GlaxoWellcome Experimental Research (GWER) and Scientific Computing (World-Wide)
Geneva, Switzerland

Ron Shamir
Professor, Department of Computer Science
Tel Aviv University
Tel Aviv, Israel

Roded Sharen
Department of Computer Science
Tel Aviv University
Tel Aviv, Israel

Seishi Shimizhu
Department of Biochemistry
Faculty of Medicine, University of Toronto
Toronto, Ontario, Canada

Ilya N. Shindyalov
Staff Scientist, San Diego Supercomputer Center
University of California, San Diego
La Jolla, California

Temple F. Smith
Professor and Director, Department of Biomedical Engineering
Boston University
Boston, Massachusetts

Victor V. Soloveyv
Director, EOS Biotechnology
South San Francisco, California

Lusheng Wang
Assistant Professor, Department of Computer Science
City University of Hong Kong
Kowloon, Hong Kong, China

Zhuozhi Wang
Department of Computer Science
University of Western Ontario
London, Ontario, Canada

Haim J. Wolfson
Associate Professor, Computer Science Department
Tel Aviv University
Tel Aviv, Israel

Limsoon Wong
Deputy Director, Bioinformatics Lab
Kent Ridge Digital Labs
Singapore

Dong Xu
Staff Scientist, Computational Biology Section
Life Sciences Division
Oak Ridge National Laboratory
Oak Ridge, Tennessee

Shizhong Xu
Associate Professor, Department of Botany and Plant Sciences
University of California, Riverside
Riverside, California

Ying Xu
Group Leader/Senior Staff Scientist, Computational Biology Section
Life Sciences Division
Oak Ridge National Laboratory
Oak Ridge, Tennessee

Kaizhong Zhang
Associate Professor, Department of Computer Science
University of Western Ontario
London, Ontario, Canada

Michael Q. Zhang
Associate Professor, Watson School of Biological Sciences
Cold Spring Harbor Laboratory
Cold Spring Harbor, New York

Index

AAT (analysis and annotation tool), 66
Ab initio approaches, 403, 467–469. *See also* Protein folding
Acceptor splice site, 216
Accuracy, 209–210, 229–230
  homology-based modeling, 458–460, 463
  prediction by similarity, 231–233
ACT binding, 75
Activators, 250
Affinity, 282
Affymetrix, 238
Agglomerative clustering, 144
Aggregation, 431–434
Algorithms. *See also* Datamining; Recognition function
  approximation, 81–82, 87–97
  AverageConsensusAlign, 93
  AverageSPAlign, 91
  Biocompress, 158–159
  BLAST, 238, 388, 391, 451
  CAST, 281–282, 286–290, 292
  center star approach, 88
  Cfact, 159–160
  CLICK, 277–278, 280–281, 286–290, 294–295
  combinatorial, 349–350
  computation volume reduction, 85–87
  DiagonalConsensusAlign, 93–95
  DiagonalSPAlign, 92–93
  DSC, 379
  DSSP, 376–379
  dynamic programming, 46–52 (*see also* Dynamic programming)
  EM, 21, 23–27, 32, 34, 162, 257–258
  energy minimization, 350–357, 453
  exact, 83–87
  FGENEH, 224, 226–227
  Fgenes, 227–230
  Fgenesh, 230, 235, 238–239
  Fgenesh+, 232–233
  Fgenesh_c, 233
  GenCompress, 158–160
  HCS, 277–280
  hierarchical clustering, 275–276
  hinge-bending, 509–511
  Hirschberg, 52–56
  k dimension programming, 84–85
  K-means, 276–277
  linear-space, 52–56
  linear time, 164
  l-star approach, 88–89
  MaxHom, 388
  MCMC, 182–185
  Metropolis-Hastings, 29–31, 183–184, 194
  multiple sequence alignment, 71–73 (*see also* Multiple sequence alignment)
  nearest-neighbor, 391–398
  neural-network based, 388–391
  NNSP, 379
  NNSSP, 393–395
  Nussinov's, 351
  PHD, 379–380, 386, 388, 391
  PromoterInspector, 263
  PROSPECT, 478–483
  PSI-BLAST, 391, 451, 457
  PSI-PRED, 380, 391
  RandomAlign, 89, 92
  reversal, 139–140, 145
  reversible jump MCMC, 180, 184–185
  self-organizing maps, 282–284
  SIM, 58
  solution assessment, 284–285
  SSP, 382–386
  SSPAL, 379–380, 394–396
  Steinerization, 146
  stochastic, 103–107, 345, 359–361
  STRIDE, 379, 398
  trace back, 50, 52, 350
  Waterman-Eggert, 394–395
  Zuker's, 350–357
Alignment. *See also* Docking; Protein folding
  approximation algorithms, 81–82
  Clustal W, 98–100
  comparative modeling, 451
  consensus, 76–77, 83, 85–87, 92–95
  content specific discrimination, 209
  covariation, 357–359
  exon-intron identification, 65–66
  frame specific discrimination, 209
  genomic comparison, 63–65
  hardness, 81–83
  iterative methods, 100
  multiple sequence, 255–259, 393 (*see also* Multiple sequence alignment)
  pairwise cost schemes, 81
  phylogenetics, 114
  position specific discrimination, 206–208
  progressive methods, 98–100
  PROSPECT, 477–478
  protein threading, 475–476
  RandomAlign, 92
  SP, 76, 82–92
  TF binding sites, 252–261
  tree, 71, 77–81, 85–87, 95–97, 100–103
All-atom models, 405–406, 414
Alleles, 111. *See also* Mapping
  epistatic effects, 195–196
  maximum likelihood estimate (MLE), 192–194
  probability model, 191–192
Alzheimer's disease, 431

Amino acids, 62–63. *See also* Protein folding
characteristics description, 381–382
DCS method, 386–388
discriminant analysis, 380–388
helices, 368–370
protein threading, 476
Ramachandran plots, 365–368
Annotation, 233–238
pathway reconstruction, 312–314
Anticodons, 202
Aperiodicity, 30–31
Approximation algorithms, 81–82
multiple sequences, 87–97
Arabidopsis, 204
Arc repressor homodimer, 429
Arrays, 270–272. *See also* Clusters
Association analysis, 320
Asymmetric measurement, 166
ATP binding, 73
Augmented model, 23–24
Auto-correlation, 387
AUTOGENE, 262
AverageConsensusAlign algorithm, 93
AverageSPAlign algorithm, 91

Bacteria, 135, 167
Band computation, 56
Basal machine, 251
Bayesian modeling, viii, 11–12, 42–44
block-motif, 37–41
CLICK, 280
datamining, 320–321, 335
EM algorithm, 24–27
empirical distribution, 185–186
epistatic effects, 195–196
frequentist approach, 14–17
joint distributions, 19–21
likelihood function, 192–194
Markov chain, 29–37, 39
MCMC, 182–185
missing data framework, 21–23
Monte Carlo algorithm, 27–32
multinomial, 32–33
parametric-statistical, 13–14
posterior distribution, 19–21
prediction programs, 227
probability model, 181–182, 191–192
score functions, 18–19
unobservables sampling, 194–195
Bend, 377–378
Bernoulli variable, 178
BESTORF, 223
β structures
DCS method, 386–388
discriminant analysis, 380–388
nearest-neighbor approaches, 391–398
neural-network-based approaches, 388–391
prediction accuracy, 383–386
protein threading, 473
SSP algorithm, 383
Binary coding, 163
Binding sites, 520
acceptor splice, 216
ACT, 75
ATP, 73
characteristics of, 512–514
datamining, 319 (*see also* Datamining)
residue detection, 514–515
residue distribution, 515–519
Biocompress algorithms, 158–159
Bioninformatics. *See* Computational molecular biology
Bionizzoni, P., 82
Bio-sequence. *See* Sequence data
Bipartitions. *See* Partitions
BLAST, 238, 388, 391, 451
Block-motif model
inhomogeneous background, 40–41
Markovian background, 39
multiple motifs, 41
BLOSUM, 46, 63, 81, 474
BOAT, 320
Boltzmann statistics, 474
Bond energy, 473–475
Bonding matrix, 349
Bootstrap method, 16
Bordetella pertussis, 313
Breakpoints, 139, 145–146
BRITE, 309
Brookhaven National Laboratory, 4
Brookhaven Protein Data Bank, 451
Bulge loops, 347–348

CAEP (classification by aggregating emerging patterns), 328–335
Calorimetric cooperativity, 434–436
Candidate ligand frame, 510
Canonical base pairs, 246–247
CART, 320, 336
Cartesian coordinates, 416
CASP, 467, 487, 495–496
CAST, 281–282, 292
case study, 286–290
Catalysts, 201
CATH, 471
CD4 genome, 63–66
C-diagonal alignment, 87–92

cDNA
exon-intron identification, 65–66
fingerprinting, 271–272
microarrays, 270
CDNA program, 161–162
C. elegans, 5, 204
Center star approach, 88
Centromeres, 137–138, 150
Cfact, 159–160
C4.5, 320
CFTR sequence, 73
CHAID, 320
Chain geometries, 413–417. *See also* Docking; Lattice models
Chan, Hue Sun, xi, 403–447, 525
Characterization, 238–242, 381–382
Character reduction, 147
Chargraff, Erwin, 3
CHARMM, 453, 473
Chips, 238–242
Chirality, 377–378
Chlamydia pneumoniae, 141
Chlamydia trachomatis, 141
Chou-Fasman distance, 393
Chromosomes. *See also* Genome
centromeres, 137–138
oncology, 150–151
polarity, 136–137
QTL mapping, 178 (*see also* Mapping)
sequence comparison, 63–65
telomeres, 137–138
Circularity, 137
alignment traces, 138–139
breakpoints, 139
edit distances, 139
reversal, 139–140
translocation, 141
transposition, 140
Cladogenesis model, 112–113
Cleansing, 318
Cleavage, 221–223
CLICK, 277–278, 280–281, 294–295
Clinical records, 328–335
Cloners, 233
Clustal W, 98–100
Clusters, 150, 269, 291–293, 295–299
agglomerative, 144
approach choice, 291
CAST, 281–282
cDNA microarrays, 270
CLICK, 277–278, 280–281
datamining, 319
HCS, 277–280
hierarchical, 275–276
KEGG, 301–315
K-means, 276–277
oligonucleotide microarrays, 270–272
quality evaluation, 291, 294
self-organizing maps, 282–284
solution assessment, 284–285
C matrix, 350
Coarse-grained statistical modeling, 406–407
Coding, 163–164. *See also* Sequence data
content specific discrimination, 209
5′-exon, 229
frame specific discrimination, 209
gene expression, 201–202
HMM-based approaches, 224–227
maximum likelihood estimate (MLE), 192–194
ORFs, 223–224, 228–229, 261, 286–290
position specific discrimination, 206–209
promoter recognition, 249–267
Collection, 318
Column cost function. *See* Cost
Combinatorial algorithm, 349–350
Combined distances, 141–142
Comparative modeling, 403, 449, 464–466
automation, 453–454
construction of model, 451–452
membranes, 462
mutations, 461
refinement, 452–453
synopsis, 450–453
template identification, 450–451, 462–463
COMPEL, 260
Compensatory base change, 357–359
Complex systems, 303
COMPOSER, 457
Compression, 157, 169–171
gain function, 159–160
GenCompress, 158–160
GTAC, 161–164
whole genome comparison, 164–168
Computational efficiency, 85–87, 115
Computational molecular biology
cluster algorithms, 269–299
datamining, 317–341
history of, 3–4
interdisciplinary nature of, vii
Conditional distributions, 31
Confidence interval, 11
Conformational propagation, 414, 431–434
CONGEN, 458
CONSENSUS, 256–257
Consensus alignment, 76–77, 92–95
computation volume reduction, 86–87
hardness, 83
Consistency, 115

Content specific discrimination, 209
Context-free grammar (CFG) method, 359–361
Cost, 75. *See also* Time
  consensus alignment, 76–77, 93–94
  graph matrix, 102
  maximum likelihood estimate (MLE), 113–114
  pairwise, 81
  SP alignment, 76
  tree alignment, 77–81
Covariation, 211–212, 357–359
CpG islands, 261–262
CPHmodel, 457
Crick, Francis, 3
Critical points, 505–506
Cross links, 489–490
Cutoffs, 60–61
Cystic fibrosis (CF) gene, 73

DALI, 487
dapC, 312–313
Data. *See also* Mapping
  augmentation, 21
  missing, 21–23, 38
  prior distribution, 17
Databases, 73. *See also* Docking
  Annotation, 233–238
  CATH, 471
  comparative modeling, 403
  COMPEL, 260
  EMBL, 202
  FSSP, 471, 486, 491
  GenBank, 5, 202–204, 219, 232–233
  HSSP, 386, 388
  InfoGene, 203–204, 212, 233–235
  KEGG, 301–315
  PDB, 376, 451–453, 472, 495–496, 508
  SCOP, 471
  SWISS-PROT, 450, 453–455
  trEMBL, 450, 453–455
  TRRD, 260
Datamining, 317, 336–341
Dayhoff, Margaret, 4
Dbscan, 232, 235–236
DCS method, 386–388
Deduction, 403
DeEP method, 334–335
Deletion. *See* Indels
DelPhi, 410
Dendograms, 275
Desolvation peak, 412
DFALIGN, 98, 100
Diagonal band, 87–92
DiagonalConsensusAlign algorithm, 93–95
DiagonalSPAlign algorithm, 92
Dielectric constant, 408, 410
Diemand, Alexander, 449–466, 525
Dirchlet distribution, 20, 32, 35, 38
Discriminant analysis
  characteristics description, 381–382
  DCS method, 386–388
  position specific, 206–209
  quadratic, 211–212, 222–223
  SSP algorithm, 382–383
Diseases, 73, 431, 433, 461
Disequilibrium, 176
Distances, 114, 138, 142, 144
  Chou-Fasman, 393
  edit, 139
  energy function, 474
  exemplar, 148–149
  Hamming, 162
  Jukes-Cantor, 124, 146
  Mahalonobis, 211–212, 215
  normalized, 127–128
  translocation, 141
  transposition, 140
Distributions
  block-motif model, 37–41
  Dirchlet, 20, 32, 35, 38
  EM algorithm, 24–27
  empirical, 185–186
  epistatic effects, 195–196
  geometrical, 226
  Gibbs sampler, 31
  iid, 13–14, 20, 37–39
  multinomial modeling, 32
  position specific, 206–209
  posterior distribution, 17–21, 24–27, 35
  quadratic analysis, 211–212
  statistical significance, 259–260
Divide and conquer strategy, 479
D. melangaster, 204
DNA, viii, ix–x. *See also* Sequence data
  block-motif model, 37–41
  cDNA microarrays, 270
  compression, 157–171
  double helix, 3
  exon-intron identification, 65–66
  function comparison, 45, 249
  gene expression, 201–202 (*see also* Genes)
  multinomial modeling, 32–33
  promoter recognition, 249–264
  repetitive patterns, 37
Docking, 503–504, 521–524
  binding epitopes, 512–520
  critical points, 505–506
  hinge-bending flexible matching, 507–511
  residue detection, 514–515

residue distribution, 515–519
rigid-body, 505–507
site characteristics, 512–514
Donor splice sites, 215–216
Doolittle, Russell, 4–5
Double helix, 3, 136–137. *See also* Helices
Doubling, 147–148, 381
Drosophila, 3, 235, 237
Drug design. *See* Docking
DSC, 379
DSSP algorithm, 376–379
Duplication, 111, 147–151
Dynamic programming, 84
Clustal W, 98–100
docking, 504
energy minimization algorithms, 350–357
Fgenes, 227–229
HMM-based approaches, 224–227
internal exon recognition, 228–229
maximum likelihood estimate (MLE), 192–194
single gene prediction, 223–224

E. coli, 4, 312
Edit distances, 139
Electrostatics, 408–410
all-atom models, 405–406
docking, 514
protein threading, 473–475
El-Mabrouk, Nadia, ix, 135–155, 525
EM algorithm, 32, 34, 162
Bayesian modeling, 21, 23–27
promoter recognition, 257–258
EMBL, 202
Emerging patterns. *See* Datamining
Empirical distribution, 185–186
Empirical force fields, 407, 414
Enchancesomes, 251
Energy function, 473–475
Energy minimization algorithm, 350
base pairs, 351
homology-based modeling, 453
loop dependent, 351–357
Enhancers, 249–250
Enrichment, 318
ENSEMBL, 203–204
Enthalpy, 434
Entropy, 106, 161–164
Environmental effects, 175
Environment class, 391–393
EOS Biotechnology, 239
Epistatic model, 195–196
Epitopes, 520–524
residue detection, 514–515
residue distribution, 515–519
site characterization, 512–514
Equations
amino acid frequencies, 390
Bayes, 257
block-motif, 38, 40–41
characteristics description, 381–382
Chou-Fasman distance, 393
computation volume reduction, 86
Dirichlet distribution, 20
EM algorithm, 25
energy function, 473–475
energy minimization algorithm, 351–357
entropy estimation, 163
environment score, 392
epistatic model, 196
Gibbs, 39
HMM, 36
joint distribution, 19, 38
least squares, 177–179
likelihood function, 20
linear model, 177
maximum likelihood estimate (MLE), 178–179, 192–194
MCMC, 29, 183
Metropolis-Hastings, 184
Monte Carlo analysis, 27, 29
nuisance parameters, 20
Poisson-Boltzmann, 410
posterior distribution, 19
potential energy, 405
probability model, 181–182, 191–192
PROSPECT, 474–475
relative information, 255
reversible jump MCMC, 184–185
SCFG, 360
statistical significance, 259
unobservables sampling, 194
weighted least squares, 179
z-score, 477
Equivalence, 61
Estimators, 14–16
ESTs, 212–213, 238
Eukaryotes, 136–137
functional signals, 206–223
gene expression, 201–202
Hannenhalli-Pevzner theory, 141–144
multiple gene prediction, 224–229
PolII promoter, 217–221
PolyA signals, 221–223
promoter recognition, 249–254
structural characteristics, 202–205
European Bioinformatics Institute, 204
Evolution, 141. *See also* Tree models
mutations, 428–431

Exact algorithms, 83
computation volume reduction, 85–87
k dimension programming, 84–85
Exemplar distances, 148–149
Exhaustive pattern search, 254–255
Exons, 57, 62, 201
5′-coding, 229
GTAC, 161–164
HMM-based approaches, 224–227
QDA, 211–212
single gene prediction, 223–224
Expectation Maximation (EM), 34, 162
Bayesian modeling, 21, 23–27
promoter recognition, 257–258
EXPRESSION, 309

FACT, 320
Fas ligand, 453
fastDNAML, 114, 119
Felsenstein zone, 119–120
Ferromagnetism, 407
FGENEH, 224, 226–227
Fgenes, 227–230
Fgenesh, 230, 235, 238–239
Fgenesh+, 232–233
Fgenesh_c, 233
Filling algorithm, 350
Fingerprinting, 271
cluster algorithms, 275–284
solution assessment, 284–285
Fisher, R. A., 3
Fisher's linear discriminant, 210–211
Fitch, Walter, 4
5′-coding exon, 229
Folding. *See* Protein folding
Force fields, 407, 414
FORTRAN, 198
Forward-backward method, 35
Frames, 209, 510
Free energy rules, 349–350, 412
Frequency matrix, 90
Frequentist approach, 14–17
Frozen approximation, 476
FSSP, 471, 486, 491
Full Automatic Modeling System, 457
Functional signals
content specific discrimination, 209
frame specific discrimination, 209
linear discriminant function (LDF), 210–211
PolII promoter recognition, 217–221
PolyA recognition, 221–223
position specific discrimination, 206–208
prediction performance measures, 209–210
quadratic discriminant analysis, 211–212
splice sites, 212–216

GAP (global alignment program), 63, 65–66
Gaps, 46
Clustal W, 98–100
Gauss' law, 408
Gaussian network model, 407
GCG, 100
GenBank, 5, 202–204, 219, 232–233
GenCompress, 158–160
GeneChips, 238
GeneCluster, 283
GeneParser, 224
Genes, 302, 307–308. *See also* Clusters; Mapping
accuracy of identification, 229–231
CF, 73
cladogenesis model, 112–113
epistatic effects, 195–196
eukaryotic, 136–137 (*see* also Eukaryotes)
evolution models, 111–115
expression steps, 201–202
functional signal recognition, 206–223
homologous, 45, 113, 139
horizontal transfer, 111
inheritance, 189–191
large-scale expression, 260
mutation, 111
physical structure, 3 (*see also* Structure)
PolII promoter, 217–221
promoter recognition, 249–264
splice sites, 212–216
Usp29, 62–63
GeneScan, 239
Gene Structures' Java Viewer, 233–235
Genie, 227
Genome, ix, 151–155, 308. *See also* Mapping
annotation, 233–238
breakpoints, 139, 145–146
CD4, 63–66
centromeres, 137–138
character reduction, 147
combined distances, 141–142
DNA sequence compression, 157–171
edit distances, 139
exemplar distances, 148–149
exon-intron identification, 65–66
Hannenhalli-Pevzner theory, 142–144
horizontal transfer, 149–150
Human Genome Project, vii, 5, 157, 237–238, 467, 494
KEGG, 301–315
linearity vs. circularity, 137
median problem, 145–146

multigene families, 138, 148–149
multiple gene prediction, 224–229
network prediction, 311–312
phylogenetic analyses, 144–147 (*see also* Phylogenetics)
polarity, 136–137
probability models, 146
protein threading, 494–497 (*see also* Protein threading)
reversal, 139–140, 145
sequence comparison, 63–65 (*see also* Sequence data)
species, 111
Steinerization algorithm, 146
synteny, 136, 141
telomeres, 137–138, 150
translocation distances, 141
transposition distances, 140
Genscan, 230
Geometry. *See also* Docking
chain, 58–61, 413–417
distribution, 226
Hashing, 514
3D assignment, 376–379, 449–450, 452
Gibbs sampling, 11, 21, 31
block-motif model, 38–40
HMM, 35–36
motif identification, 104–107
promoter recognition, 258–259
unobservables, 194–195
Gilbert, Wally, 4
Global alignment, 45
band computation, 56
dynamic programming algorithm, 46–52
GAP, 63
linear-space algorithm, 52–56
Globular proteins, 365–367
β structures, 370–373
β turns, 373–375
fold templates, 472
helices, 368–370
Poisson-Boltzmann approach, 409
Glycine, 366
Go models, 418–419
GOR III, 386
Goto, Susumu, x, 301–315, 525
Graphs, 101–103. *See also* Mapping
KEGG, 304–305, 310–314
Ramachandran plots, 365–367
GRASP, 410
Green Plant Phylogeny, 130
GROMOS, 453
Group table, 308–309
GTAC program, 161–164
Guex, Nicolas, 449–466, 525
Gusfield, D., 88

Hairpin loops, 347–348
Haldane, J. B. S., 3
Hamming distance, 162
Hannenhalli-Pevzner theory, 141–144
MAX SNP, 107
MQC, 122–123
NP, 81–82
PTAS, 82
SP alignment, 82–83
Traveling Salesman Problem, 145–146
HCS, 270–280
Helices, 348
α, 365, 368–370, 376–378
DCS method, 386–388
discriminant analysis, 380–388
nearest-neighbor approaches, 391–398
neural-network-based approaches, 388–391
prediction accuracy, 383–386
protein threading, 473–475
secondary structure, 368–370
SSP algorithm, 382–383
3D structure assignment, 376–379
Heuristic approach, 97
fastDNAML, 114
iterative methods, 100
progressive alignment, 98–100
sequence graphs, 101–103
stochastic algorithms, 103–107
HEXON, 223
Hidden Markov model (HMM), 40, 321
Bayesian modeling, 33–37
Hidden semi-Markov model (HSMM), 37
Hierarchical clustering, 275–276
H. influenzae, 313
Hinge-bending flexible matching, 507–509
Hirschberg algorithm, 52–56
Homogeneity, 269
Homology-based modeling, 45, 449, 464–466
membranes, 462
mutations, 461
refinement, 452–453
template identification, 450–451, 462–463
Horizontal transfer, 149–150
HP+ models, 419–422, 437
H. pylori, 167
HSSP, 386, 388
HTG sequences, 213
Huang, Xiaoqiu, viii, 45–69, 525
Human Genome Project, vii, 5, 157, 467, 494
Hybrid-210 system, 416
Hybrids, 189–191

Hydrophobic interactions, 410–413
 docking, 514
 moment characteristic, 382
Hypercleaning, 127–130

ID3, 320
ID5, 320
If-then filtering, 387–388
imid position, 52–56
Inbred lines, 176
Indels (insertions/deletions)
 global alignment, 46–52
 linear-space algorithm, 52–56
 SP alignment, 93
Independent and identically distributed (iid) model, 13–14, 20
 block motif model, 37–39
Inference, 119–120
 Bayesian, 182–185
 frequentist approach, 14–17
 statistical modeling, 13–17
InfoGene, 203–204, 212, 235
 annotation, 233–234
Inheritance. *See* Alleles
Initial state probability, 226
Insertion. *See* Indels
Interaction schemes, 417–418
Interior loops, 347–348
Internet. *See* Web resources
Introns, 57, 62, 201
 accuracy of identification, 229–231
 GTAC, 161–164
 HMM-based approaches, 224–227
 splice sites, 212–216
Inversion, 28, 139–140
Ions, 405–406, 408–410. *See also* Electrostatics
Irreducibility, 30

Jaccard coefficient, 284–285
Jackknife method, 16, 383
JEP method, 334
Jiang, Tao, viii–ix, 71–110, 525
jmid position, 52–56
Joint distributions, 19–21
 augmented model, 23–24
 quantitative traits, 175 (*see also* Quantitative traits)
Jukes-Cantor distance, 124, 146

Kanehisa, Minoru, x, 301–315, 525
Karyotypes, 150–151
Kaya, Huseyin, 403–447, 525
k dimensional programming, 84–85
Kearney, Paul, ix, 111–133, 525

KEGG (Kyoto Encyclopedia of Genes and Genomes), x, 301–302, 315
 annotation, 312–314
 BRITE, 309
 complex systems, 303
 GENES, 307–308
 GENOME, 308
 LIGAND, 309–310
 network prediction, 311–312
 PATHWAY, 306–307
Kendrew, John, 3
Kent Ridge Digital Labs, 317
Kinetic energy. *See also* Protein folding
 conformational propagation, 431–434
 non-Arrhenius, 436
 PMF, 408–413
K-loop decomposition, 347–349
 energy minimization, 351–357
K-means, 276–277, 292, 294–295
Knobs, 505
Knots, 347
Knowledge-based approaches, 403–404. *See also* Datamining
Koetzle, Tom, 4
Kolmogorov complexity, 164–165

Lactose operator, 4
Latent-class model, 22
Lattice models
 aggregation, 431–434
 calorimetric cooperativity, 434–436
 chain geometries, 414–417
 conformational propagation, 431–434
 HP+ models, 419–422
 interaction potentials, 417–427
Least squares fitting, 144
 QTL mapping, 176–179
Leucine, 366
Li, Ming, ix, 157–171, 525
Lifted tree, 95
LIGAND, 302, 309–310
Ligands
 docking, 505–512
 Fas, 453
 hinge-bending, 507–511
Likelihood function, 15
 EM algorithm, 24–27
 HMM, 33–37
 profile, 16, 42, 90
Linear discriminant function (LDF), 210–211, 219, 229, 380–381
Linearity, 137
 alignment traces, 138–139
 breakpoints, 139

combined distances, 141–142
edit distances, 139
reversal, 139–140
space, 52–56
translocation, 141
transposition, 140
Line crosses. *See* Bayesian mapping
Liu, Jun S., viii, 11–44, 525
Loaded tree, 95
Local alignment, 56–58
Locus control regions (LCRs), 249
Loops
bulge, 347–348
energy minimization algorithm, 351–357
hairpin, 347–348
non-conserved, 452
Los Alamos National Laboratory, 5
Low-degree vertices, 279–280
l-star approach, 88–89

Ma, Buyong, xii, 503–525
Mahalonobis distances, 211–212, 215
Mapping, 3
Bayesian, 181–196 (*see also* Bayesian modeling)
disequilibrium, 176
maximum likelihood estimate (MLE), 192–194
MCMC, 182–185
mixed model, 189–191
mutations, 428–431
neural networks, 486
probability model, 191–192
QTL, 175–176 (*see also* Quantitative trait loci (QTL))
Ramachandran plots, 365–368
self-organizing, 282–284, 286–290, 292, 294–295
Marginal mode, viii
Margoliash, Emanuel, 4
Markers, 175
Bayesian mapping, 181–186
least squares method, 176–179
maximum likelihood estimate (MLE), 192–194
Markov chain, viii
Bayesian mapping, 182–185
content specific discrimination, 209
distributions, 20
HMM-based approaches, 225
homogenous model, 33
Monte Carlo method, 29–32
position specific discrimination, 208
reversible jump MCMC algorithm, 184–185
Markov model
block-motif model, 39
hidden, 33–37, 40, 224–227, 321
homogeneous, 33
Mass spectrometry, 469
Matching, 507–511
MATLAB, 286
Matrices, 47
BLOSUM, 81
bonding, 349
combinatorial algorithm, 349–350
CONSENSUS, 256–257
covariation, 211–212
energy function, 473–475
epistatic effects, 195–196
functional signals, 206–223
global alignment, 46–56
motif identification, 106
mutation, 392
PAM, 81
similarity, 273
substitution, 393
TF binding sites, 252–261
weighted, 216, 218, 252, 273, 278–280
Maxam, Allan, 4
MaxHom, 388
Maximum likelihood estimate (MLE), 15–17
Bayesian mapping, 192–194
genome rearrangement, 144
homogenous Markov model, 33
missing data formulation, 21
multinomial modeling, 32
phylogenetics, 113–114
QTL mapping, 178–179
quartet methods, 117, 121–123
Maximum parsimony method, 114, 120, 144
Maximum quartet consistency (MQC), 117, 121–123
MAX SNP-hard, 107
Mean force potentials, 407
electrostatics, 408–410
hydrophobic interactions, 410–413
Median problem, 145–146
Membrane proteins, 462
MEME, 258
Mendelian inheritance, 175, 194
Mesoscopic length, 407
Metropolis-Hastings algorithm, 29–31, 183–184, 194
MFOLD, 351
MHC-binding peptide, 317–319, 335
short, 322–328
Mice, 204
Midposition, 52–56
Minkowski measure, 284
Missing data formulation, 21–23, 38
Mixed model, 189–191
Model fitting, 14

MODELLER, 451, 457, 487–488
Molecular surface variability. *See* Docking
Monte Carlo analysis, viii
 Bayesian mapping, 182–185
 Gibbs sampler, 31
 inversion method, 28
 Markov chain, 29–32
 rejection method, 28–29
 reversible jump MCMC algorithm, 184–185
 simple, 27–28
Morgan, T. H., 3
Motifs
 conserved, 71
 Gibbs sampling, 104–107
 Identification, 103–107
 TF binding sites, 252–261
Move sets, 437
mRNA
 gene expression, 201–202
 splice sites, 212–216
Muller, Hermann, 3
Multigene families, 138, 147
 duplication, 150–151
 exemplar distances, 148–149
Multinomial distributions, 20, 32
 protein threading, 496–497
Multiple loops, 347, 349
Multiple sequence alignment, 71–72, 108–110
 approximation algorithms, 81–82, 87–97
 computation volume reduction, 85–87
 consensus, 76–77, 92–95
 diagonal band, 87–88
 exact algorithms, 83–87
 heuristic approaches, 97–107
 k dimension programming, 84–85
 l-star approach, 88–89
 pairwise cost schemes, 81
 PTAS, 89–92, 97
 sequence graph approach, 101–103
 SP, 76, 87–92
 stochastic algorithms, 103–107
 tree, 77–81, 95–97, 100–101
Mutation matrix, 392
Mutations, 3, 111, 461
Mutual algorithmic information, 165
Myoglobin, 239
MZEF, 223

National Institute of General Medical Sciences, 494
Nearest-neighbor approaches, 391–398
Neighborhood recovery, 128–129
Neighbor joining, 114, 144
Nematodes, 62
Neoplastic patterns, 151
Neural networks
 protein folding, 428–431
 protein secondary structure, 388–391
 threading normalization, 484–487
 training of, 486
Newton-Raphson's method, 34
NIH Structural Genomics Initiative, 494–495
NMR, 450, 453, 467, 469
 intra-molecular cross-links, 489–490
NNSP, 380
NNSSP, 393–394
Nodes, 49–50
NOEs, 488–494
Non-histon proteins (NHP), 251
Non-parametric approach, 13–14
Normalization, 484–487
Normalized distance, 127–128
NP-completeness, 140
NP-hardness, 81–82. *See also* Hardness
Nuisance parameters, 16, 20
Null model, 14
Nussinov, Ruth, xii, 503–525
Nussinov's algorithm, 351

Observed-data likelihood, 23
Observables, 197–198
 mixed model, 189–191
 probability model, 191–192
Oligonucleotide microarrays, 270–272
Oncology, 150–151
Optimal alignment, 50
 band computation, 56
 linear-space algorithm, 52–56
 local, 56–58
Optimal parse, 226–227
Optimization models, 73–75
 approximation algorithms, 81–82
 consensus alignment, 76–77
 pair-wise cost schemes, 81
 SP alignment, 76
 tree alignment, 77–81
ORFs (orthologous coding regions), 228–229, 261
Ortholog group table, 308–309
Outbred populations, 176
Outlier analysis, 319
OVER, 396

Painter, T. S., 3
Pairs, 59–61, 71
 all-atom models, 405–406
PAM, 46, 81, 473–474
Parametric modeling
 all-atom models, 414

Bayesian model, 12, 180–181
empirical force-field, 407
frequentist approach, 14–17
least squares method a, 176–179
nuisance parameters, 20
QDA, 211–212
statistical modeling, 13–14
Parse probability, 226–227
Partitions, 272–274. *See also* Algorithms
bipartitions, 127–130
CAST, 281–282
CLICK, 277–278, 280–281
HCS, 277–280
K-means, 276–277
self-organizing maps, 282–284
PATHWAY, 302, 306–307
Pathway reconstruction, 312–314
Patterns. *See* Datamining
Pauling, Linus, 4, 369
Pedigrees, 188
dominance, 195–196
epistatic effects, 195–196
maximum likelihood estimate (MLE), 192–194
mixed model, 189–191
probability model, 191–192
unobservables sampling, 194–195
Peitsch, Manuel C., xii, 449–466, 525
Penalty rule, 121
Peptides
β structure, 370–375
chain geometries, 413–417
MHC-binding, 317–319, 322–328, 335
short, 322–328
Performance ratio, 82
Permutation, 350
Perturbation, 30
PHD, 379–380, 386, 388, 391
Phenotypic distribution, 175
Phylogenetics, ix
assessment, 114–115
character reduction, 147
comparative methods, 357–359
evolution models, 111–115
footprinting, 261
genome rearrangement, 144–147
maximum likelihood estimate (MLE), 113–114
median problem, 145–146
probability theory, 146
quartet methods, 115–131
Steinerization algorithm, 146
trees, 4, 77–78
Phylogenetic resources, 131
Pima Indians, 328–335
PMF (potentials of mean force), 408–413
Point mutations, 111
Poisson-Boltzmann approach, 408–410
Polarity, 136–137
PolII promoters, 217–221
PolyA, 201, 221–223, 228
Polyadq program, 222–223
POLYAH program, 222–223
Polygenic traits. *See* Quantitative traits
Polymer models, 413
chain geometries, 414–417
Go models, 418–419
HP+ models, 419–422
interaction potentials, 417–427
lattice representation, 414–437
Polymorphic molecular markers, 175
Polynomial time approximation scheme (PTAS), 82, 107
DiagonalConsensusAlign algorithm, 93–95
SP alignment, 89–92
tree alignment, 95, 97
Positional cloners, 233
Position specific discrimination, 206–209
Posterior distribution, 17–21
EM algorithm, 24–27
HMM, 35
Potential energy, 405–407. *See also* Lattice models
electrostatics, 408–410
hydrophobic interactions, 410–413
PMF, 408–413
protein threading, 473–475
Prediction, 223–224, 399–401. *See also* Datamining
ab initio, 468
annotation, 235, 237
CAEP, 328–335
combinatorial algorithm, 349–350
discriminant analysis, 227–229, 380–388
docking, 503–504 (*see also* Docking)
energy minimization algorithm, 350–357
globular proteins, 365–365
homology-based modeling, 449–456 (*see also* Homology-based modeling)
nearest-neighbor approaches, 391–398
neural networks-based approaches, 311–312, 388–390
OVER, 396
phylogenetic comparison, 357–359
PROSPECT, 477–478
RNA secondary structure, 345–361
stochastic context-free grammar method, 359–361
3D structure assignment, 376–379
TSS, 261–264
UNDER, 396
WRONG, 396

Preinitation complex (PIC), 249–250
Primordial folds, 471
Principle of repeated sampling, 15
Prion diseases, 431, 433
Prior distribution, 17–18
Probability, 11, 191–192. *See also* Prediction
 Bayesian modeling, 181–182, 191–192 (*see also* Bayesian modeling)
 CLICK, 280–281
 content specific discrimination, 209
 EM, 257–258
 frame specific discrimination, 209
 frequentist approach, 14–17
 initial state, 226
 linear discriminant function, 210–211
 multiple gene prediction, 224–229
 position specific discrimination, 206–209
 quadratic discriminant analysis, 211–212
 SCFG, 359–361
 single gene prediction, 223–224
 splice site recognition, 215–217
 TSSW program, 217–221
Profile likelihood, 16, 42, 90
Prokaryotes, 137, 141
Proline, 366
PromoterInspector, 263
Promoter recognition, 201, 249–251, 265–267
 exhaustive pattern search, 254–255
 large-scale expression, 260
 multiple sequence alignment, 255–259
 regulatory module construction, 260
 statistical significance, 259–260
 transcription factor binding sites, 252–255
 TSS prediction, 261–264
PROMOTERSCAN, 262
PROSPECT, 469, 477
 energy function, 474–475
 score normalization, 484–487
 structure prediction, 487–488
Protein Data Bank (PDB), 376, 451–453
 fold templates, 472
 protein threading, 495–496
Protein folding, 438–447
 aggregation, 431–434
 all-atom models, 405–406, 414
 calorimetric cooperativity, 434–436
 chain geometries, 414–417
 conformational propagation, 431–434
 electrostatics, 408–410
 Go models, 418–419
 HP+ models, 418–422, 437
 hydrophobic interactions, 410–413
 interaction potentials, 417–427
 kinetics, 436–437
 knowledge-based approach, 403–404
 lattice representation, 414–417, 427–437
 mean force potentials, 407–413
 model constraints, 434–436
 polymer models, 413–417
 solvation effects, 407–413
 statistical mechanics models, 406–407
Proteins, 3, 399–401
 docking, 506 (*see also* Docking)
 families, 470–471
 folding, 380
 globular, 365–375, 409, 472
 homology-based modeling, 449–456 (*see also* Homology-based modeling)
 KEGG, 301–315
 membrane, 462
 superfamilies, 470–471
 3D structure assignment, 376–379
Protein threading, 467–469
 alignment, 475–476
 assessment, 476–477
 energy function, 473–475
 fold templates, 470–472
 intra-molecular cross-links, 489–490
 local, 495
 mini, 495–496
 multiple-sequence data, 496–497
PROSPECT system, 477–478
PSI-BLAST, 391, 451, 457
PSI-PRED, 380, 391
Pseudo-knots, 349
PUBLIC, 320
Puzzling, 120–121
p-value, 11, 477

Q-function, 34
Quadratic discriminant analysis (QDA), 211–212
Quadratic discriminant function (QDF), 212, 222–223
Quantitative trait loci (QTL), x, 175
 Bayesian mapping, 180–196
 least squares, 176–179
 maximum likelihood estimate (MLE), 178–179
Quantitative traits, 17–18, 197–199
 Bayesian mapping, 181–196
 empirical distribution, 185–186
 epistatic effects, 195–196
 likelihood function, 192–194
 MCMC mapping, 182–185
 mixed model, 189–191
 probability model, 181–182, 191–192
 QTL mapping, 175–181
 unobservables sampling, 194–195

Quartet methods, 115–116
bipartition support, 127–130
hypercleaning, 127–130
maximum consistency, 122–123
neighborhood recovery, 128–129
semi-definite programming, 123–125
taxonomic sampling, 119–120
topological inference, 119–120
QUEST, 320

Ramachandran plots, 365–368
RandomAlign algorithm, 89, 92
Randomness, 17
block-motif model, 39
genetic drift, 111
Gibbs sampler, 31
SOM, 282–284
Ranking problem, 511–512
Receptors. *See* Docking
Recognition function
annotation, 233–238
characterization/verification, 238–242
internal exon, 228–229
ORF, 223–224
PolII promoter, 217–221
PolyA signals, 221–223
prediction by similarity, 231–233
promoter, 249–250 (*see also* Promoter recognition)
single gene prediction, 223–224
splice sites, 215–217
TSSW program, 217–221
Recombination, 81, 116
Reconciliation, 149
Recursion, 19, 42
HMM, 35–36
Reduction, 147
Regulatory regions, 249, 260
Rejection method, 28
Relative entropy, 106
Relative information (RI), 255
Remodeling, 251
Representation theory, 152–155
alignment traces, 138–139
breakpoints, 139, 145–146
centromeres, 137–138
character reduction, 147
combined distances, 141–142
edit distances, 139
exemplar distances, 148–149
gene order, 136–137
Hannenhalli-Pevzner theory, 142–144
horizontal transfer, 149–150
linearity vs. circularity, 137
median problem, 145–146
multigene families, 138, 148–149
phylogenetic analyses, 144–147 (*see also* Phylogenetics)
polarity, 136–137
probability models, 146
reconciliation, 149
reversal, 139–140, 145
Steinerization algorithm, 146
synteny, 136, 141
telomeres, 137–138
translocation distances, 141
transposition distances, 140
Repressors, 250
Residues, 71, 366, 370
distribution of, 515–519
Reversal, 139–140, 145
Reversible jump MCMC algorithm, 180, 184–185
Ribosomal Database Project, 112n1, 131
RNA, xi, 157
canonical base pairs of, 346
covariation, 357–359
k-loop decomposition, 347
primary structure of, 346
probabilistic models for, 11
secondary structure prediction, 345–361
3′-processing site, 221–223
Robustness, 115
rRNA, 157

Salamov, Asaf, 242
Sanger, Frederick, 3–4
Sankoff, David, ix, 135–155, 525
SARF, 486
SAS, 475
Scaffold/matrix attachment regions (S/MARs), 249
S. cerevisiae, 204
Schwede, Torsten, 449–466, 525
SCOP, 471
Score functions, 18, 49–50
chains, 59–61
docking, 503 (*see also* Docking)
PolII promoter, 217–221
PROSPECT, 479–480
SP alignment, 84–85
TF binding sites, 252–261
threading normalization, 484–487
tree alignment, 85
z-score, 476–477
SDSC1, 457
Secondary structure. *See also* Structure
β structures, 370–377
discriminant-analysis approaches, 380–388

Secondary structure (cont.)
helices, 368–370
nearest-neighbor approaches, 391–398
neural networks-based approaches, 388–391
Ramachandran plots, 365–368
RNA prediction, 345–351
3D assignment, 376–379
Seledtsov, Igor, 233, 242
Self-organizing maps (SOM) Semi-definite programming (SDP), 123–125
Sensitivity, 209–210
Separation, 269
Sequence data, 69, 157. *See also* Structure
annotation of, 233–238
CFTR, 73
characterization/verification, 238–242
comparison pertinence, 138
compositional analysis of, 32–37
DNA, 6 (*see also* DNA)
dual comparison algorithms, 61–62
dynamic programming algorithm, 46–52
exon-intron boundaries, 65–66
frame specific discrimination, 209
functional signals, 206–223
GenCompress, 158–160
genomic comparison, 63–65
ORF, 223–224
PolyA signals, 221–223
position specific discrimination, 206–209
prediction by similarity, 231–233 (*see also* Prediction)
profile methods, 468
promoter recognition, 249–264
PROSPECT, 477–478
protein comparisons, 62–63 (*see also* Proteins)
splice sites, 212–216
whole genome comparison, 164–168
Shamir, Ron, x, 269–299, 526
Sharan, Roded, x, 269–299, 526
Shimizu, Seishi, 403–447, 526
Shindyalov, Ilya N., xi, 365–401, 526
Significance level, 11
Silencers, 250
Silicon Graphics Inc., 454
SIM algorithm, 58
Single-stranded region, 349
Singletons
adoption, 279
characteristic, 381
PROSPECT, 479–480
protein threading, 473–474
SLIQ, 320
Slow-mixing, 32
Smith, Temple F., viii, 3–8, 526
Softberry, Inc., 205, 237–238
Solovyev, Victor V., x–xi, 201–248, 365–401, 526
Solvation effects, 407
electrostatics, 408–410
hydrophobic interactions, 410–413
SORFIND, 223
SP alignment, 76
computation volume reduction, 86
diagonal band, 87–92
Specificity, 209–210
Splice sites, 212–216
SPL program, 216
SPRINT, 320
SSP, 382–386
SSPAL, 379–380, 394–396
Stacked pairs, 347
Statistical modeling, 11. *See also* Bayesian modeling; Probability
coarse-grained, 406–407
energy function, 474
frequentist approach, 14–17
parametric, 13–14
protein folding, 403–404 (*see also* Protein folding)
Steinerization algorithm, 146
Stochastic algorithms
context-free grammar method, 345, 359–361
motif identification, 103–107
STRIDE, 379, 398
Strings, 61–62
Structural Genomics Initiative, 494–495
Structure, 399–401. *See also* Genome; Helices
ab initio prediction, 468
DNA, 3 (*see also* DNA)
docking, 514–519 (*see* also Docking)
energy minimization algorithm, 350–357
entropy estimation, 164
eukaryotic genes, 202–205
functional signals, 206–213
function comparison, 45
globular proteins, 365–375, 409, 472
HMM-based approaches, 224–227
homology-based modeling, 449–456 (*see also* Homology-based modeling)
k-loop decomposition, 347
nearest-neighbor approaches, 391–398
neural networks-based approaches, 388–390
PROSPECT, 487–488
RNA, 346–349 (*see also* RNA)
stochastic context-free grammar method, 359–361
Structural Genomics Initiative, 494–495
suboptimal, 357
tertiary, 345, 347, 467–469
TF binding sites, 252–261
3D assignment, 376–379

Substitution, 46, 111, 393
Super sampling, 120
Support vector machines (SVMs), 322
SWISS-MODEL, 457
SWISS-PROT, 450, 453–455
Symmetry, 30
Synteny, 136, 141
Synthesis, 201–202
Systematic amyloidoses, 431
Systematic scan Gibbs sampler, 31

TATA box, 217–221, 250, 252
Taxonomic sampling, 119–120
T-cells, 321
Telomeres, 137–138, 150
Temperature. *See also* Kinetic energy
  calorimetric cooperativity, 434–436
  protein folding, 408–409, 412–413
Templates, 450–452
  ab initio approaches, 467–469
  fold, 470–472
  PROSPECT, 477–478
  selection sensitivity, 462–463
  topological complexity, 482–483
Tertiary structures, 345, 347, 467–469
Thermolysine, 370
3D-JIGSAW, 457
3′-processing site, 221–223, 228
Threshold traits. *See* Quantitative traits
Time complexity
  ab initio approaches, 469
  all-atom models, 406
  energy minimization algorithm, 357
  exact algorithms, 83–87
  GAP, 63, 65
  hypercleaning, 129
  sequence alignment, 50, 52
TOPITS, 475
Topology and quartet topology
  fold templates, 470–472
  genome rearrangement, 144–147
  hypercleaning, 127–130
  interweighted, 117
  neighborhood recovery, 128–129
  short quartet method, 125–127
Toxicity, 321
Trace back algorithm, 50, 52, 350
Transcriptional start site (TSS), 249
  CpG islands, 261–262
  k-tuples, 263–264
  TF site scan, 262
Transcription factor (TF)
  finding, 252–254
  gene expression, 201–202
  promoter recognition, 249–254
  TSS prediction, 261–264
TRANSFAC identifier, 219
Transformation, 165
Transition, 33, 226–227
Translation, 201–202
Translocation, 141–144
Transparency, 115
Transposition, 140
Traveling Salesman Problem (TSP), 145–146
Tree alignment, 71, 77–78, 95–96, 100
  computation volume reduction, 86–87
  PTAS, 97
  sequence graphs, 101–103
TreeBASE, 131
Tree models, 113
  entropy estimation, 164
  genome rearrangement, 144–147
  quartet methods, 115–121, 125–127
  reconciliation, 149
  semi-definite programming, 123–125
  topological accuracy, 114–115
Tree of Life, 130
TreeView, 276
trEMBL, 450, 453–455
Trend analysis, 319–320
Triangle inequality, 165
Triplet frame, 510
TRRD, 260
TSSW program, 217–221
Twenty-letter models, 422–427

Ubiquitin carboxyl-terminal hydrolase, 62–63
UCLA-DOE Structure Prediction Server, 475
Ultra-310 system, 416
UNDER, 396
Uniform lifting, 95–97
Unique genes hypothesis, 138, 148–149
Unobservables, 194–195
Usp29 gene, 62–63

Vacuum permitivity, 405, 408
Valine, 366
van der Waals energy, 473–475
Vedova, G. Della, 82
Veil, 227
Verification, 238–242
Vertices, 279–280
Viruses, 138
Voting scheme, 509–510

Wang, Lusheng, 71–110, 526
Wang, Zhuozhi, 345–363, 526
Waterman-Eggert algorithm, 394–395

Watson, James, 3
Web resources
CLICK, 280
Cluster, 276
COMPOSER, 457
CONGEN, 457
CPHmodel, 457
distance matrix, 166
ENSEMBL, 204
FAMS, 457
GeneCluster, 283
InfoGene, 203, 233
MEME/MAST, 258
MODELLER, 457
PDB, 472
phylogenetics, 130–131
PromoterInspector, 263
PROSPECT, 487
SDSC1, 457
secondary structure prediction, 398
Softberry, Inc., 205, 237–238
STRIDE, 398
3D-JIGSAW, 457
TreeView, 276
TSS prediction, 262
WHATIF, 457
Weighted least squares method, 179
Weighted matrices, 216, 218
HCS, 278–280
TF binding sites, 252
Weight functions, 206–209
WHATIF, 457
Wolfson, Haim J., xii, 503–524, 526
Wong, Limsoon, x–xi, 317–341, 526
WORDUP, 255
Wright, Sewell, 3
WRONG, 396

X-rays, 4, 345, 434
protein structure, 450, 467, 469
Structural Genomics Initiative, 494
Xu, Dong, 467–502, 526
Xu, Shizhong, 175–199, 526
Xu, Ying, 467–502, 526

Yeast, 5, 62, 227
linearity vs. circularity, 137

Zhang, Kaizhong, 345–363, 526
Zhang, Michael Q., 249–267, 526
z-score, 476–477
Zuckerkandl, Emile, 4
Zuker's algorithm, 350
energy minimization, 351–357
MFOLD, 351

Frontiers of Science and Technology for the 21st Century

Editor in Chief Guo-kui Liu

Progress in Transmission Electron Microscopy

I. Concepts and Techniques

II. Applications in Materials Science

Editors Xiao-Feng Zhang, Ze Zhang

(has been published)

Handbook of Nanophase and Nanostructured Materials

Synthesis

Characterization

Materials Systems and Applications (I)

Materials Systems and Applications (II)

Editor in chief Zhong-Lin Wang

Editors Yi Liu, Ze Zhang

(has been published)

Functional Thin Films New Concepts and Technologies

Editor Dong-Lu Shi

(has been published)

Current Topics in Computational Molecular Biology

Editors Tao Jiang, Ying Xu, Michael Q. Zhang

ISBN 7-302-05378-2

9 787302 053781

ISBN 0-262-10092-4

9 780262 100922

定价：70.00元